Plenge - Szczepanowski

Simon's Basic op de Commodore 64

Een oefenboek

Commodore Bibliotheek 5
DATA BECKER
NEDERLANDS *

Simon's Basic

Plenge - Szczepanowski

Simon's Basic op de Commodore 64

Een oefenboek

Commodore Bibliotheek 5

DATA BECKER
NEDERLANDS *

Oorspronkelijke titel: Das Trainingsbuch zum Simon's Basic
Copyright © 1984 DATA BECKER GmbH, Düsseldorf
Vertaling: Chinterpret, Groningen
Copyright © 1984 A.W. Bruna & Zoon, Utrecht
Redaktie: ProCread, Groningen
Druk: Vonk Zeist
ISBN 90 229 3327 X
De boeken van DATA BECKER/NEDERLANDS* zijn gezet op een Brother CE50 met een interface van Rob Koelmans Micro Randapparatuur in Groningen
3e druk augustus 1986

De keuze en de opbouw van de programma's die in dit boek voorkomen, zijn gebaseerd op hun educatieve waarde. Alle programma's zijn getest op hun juiste werking. De uitgever kan geen enkele verantwoordelijkheid voor eventueel optredende fouten aanvaarden.

Uit deze uitgave mag niets worden openbaar gemaakt en/of verveelvoudigd door middel van druk, fotokopie, microfilm of op welke andere wijze dan ook zonder voorafgaande schriftelijke toestemming van de uitgever.

Boeken en programma's van

DATA BECKER
NEDERLANDS*

worden in de handel gebracht door:
A.W. Bruna & Zoons Uitgeversmij b.v.,
Postbus 8411, 3503 RK Utrecht
en
A.W. Bruna & Zoon n.v.,
Antwerpsesteenweg 29a, 2630 Aartselaar.

1.	**INLEIDING**	10
2.	**PROGRAMMEERHULP**	12
2.1.	Het definiëren van de functietoetsen	12
2.1.1	KEY	13
2.1.2	DISPLAY	16
2.1.3	Testopdrachten	17
2.2	Gemakkelijke programmasamenstelling	18
2.2.1	AUTO	18
2.2.2	RENUMBER	20
2.2.3	MERGE	23
2.2.4	Testopdrachten	26
2.3	LIST-hulpmiddelen	27
2.3.1	PAGE	28
2.3.2	OPTION	29
2.3.3	DELAY	30
2.3.4	FIND	31
2.3.5	Testopdrachten	34
3.	**DE BEHANDELING VAN FOUTEN**	36
3.1	Externe foutenbehandeling	37
3.1.1	TRACE	37
3.1.2	RETRACE	39
3.1.3	DUMP	40
3.1.4	COLD	41
3.1.5	OLD	42
3.1.6	Testopdrachten	43
3.2	Interne foutenbehandeling	44
3.2.1	ON ERROR	44
3.2.2	NO ERROR	46
3.2.3	ON ERROR/NO ERROR in detail	47
3.2.4	OUT	52
3.2.5	Testopdrachten	53
4.	**BESCHERMING VAN HET PROGRAMMA**	55
4.1	DISAPA	55
4.2	SECURE	56
5.	**STRUCTUUR VAN HET PROGRAMMA**	58
5.1	Logische operaties	58
5.1.1	IF...THEN...ELSE	58
5.1.2	RCOMP	60
5.1.3	REPEAT...UNTIL	61
5.1.4	LOOP...EXIT IF...END LOOP	63
5.1.5	Testopdrachten	65
5.2	Behandeling van de sprong	66
5.2.1	PROC	66
5.2.2	END PROC	67
5.2.3	CALL	68

5.2.4	EXEC	69
5.2.5	CGOTO	70
5.2.6	Testopdrachten	71
6.	**VARIABELEN**	73
6.1	LOCAL	73
6.2	GLOBAL	74
6.3	Testopdrachten	75
7.	**DE BEHANDELING VAN DE GETALLEN**	76
7.1	Rekenkundige operatoren	76
7.1.1	MOD	76
7.1.2	DIV	77
7.1.3	FRAC	79
7.1.4	EXOR	80
7.1.5	Testopdrachten	81
7.2	De omzetting van getallen	82
7.2.1	%	82
7.2.2	$	83
7.2.3	Testopdrachten	84
8.	**STRINGOPERATIES**	86
8.1	INSERT	86
8.2	INST	88
8.3	PLACE	90
8.4	DUP	98
8.5	Testopdrachten	99
9.	**CONTROLE VAN DE OUTPUT**	102
9.1	Geformateerde uitvoer	102
9.1.1	CENTRE	103
9.1.2	USE	105
9.1.3	AT	106
9.1.4	LIN	108
9.1.5	PAUSE	109
9.1.6	Testopdrachten	110
9.2	Het knipperen van het beeldscherm	111
9.2.1	FLASH	111
9.2.2	OFF	113
9.2.3	BFLASH	114
9.2.4	BFLASH 0	115
9.2.5	Testopdrachten	117
9.3	De achtergrond- en vensterkleur	118
9.3.1	COLOUR	120
9.3.2	BCKGNDS	122
9.3.3	NRM	126
9.3.4	Testopdrachten	127
9.4	Sturing: het beeldscherm	127
9.4.1	FCHR	129

9.4.2	FCOL	132
9.4.3	FILL	135
9.4.4	MOVE	136
9.4.5	INV	138
9.4.6	Testopdrachten	140
9.5	Beeldscherm scrollen	141
9.5.1	LEFT	141
9.5.2	RIGHT	143
9.5.3	UP	145
9.5.4	DOWN	147
9.5.5	Testopdrachten	148
10.	**INPUT-CONTROLE**	150
10.1	FETCH	150
10.2	INKEY	151
10.3	RESET	153
10.4	Testopdrachten	154
11.	**IN- EN UITVOERINSTRUCTIES**	156
11.1	Diskette-instructies	156
11.1.1	DISK	156
11.1.2	DIR	157
11.1.3	SCRSV	159
11.1.4	SCRLD	161
11.1.5	Testopdrachten	162
11.2	Printinstructies	162
11.2.1	COPY	163
11.2.2	HRDCPY	163
11.2.3	Testopdrachten	164
12.	**GRAFIEK**	165
12.1	Hardwarefaciliteiten	165
12.2	Grafiekmodus/kleurbepaling	167
12.2.1	HIRES	171
12.2.2	MULTI	173
12.2.3	NRM	176
12.2.4	CSET 2	177
12.2.5	LOW COL	178
12.2.6	HI COL	181
12.2.7	Testopdrachten	184
12.3	Instructies voor de grafiek	185
12.3.1	PLOT	186
12.3.2	TEST	187
12.3.3	LINE	191
12.3.4	REC	194
12.3.5	BLOCK	198
12.3.6	CIRCLE	201
12.3.7	ARC	205
12.3.8	ANGL	210

12.3.9	PAINT	214
12.3.10	Testopdrachten	217
12.4	Zelf figuren tekenen	129
12.4.1	DRAW	219
12.4.2	ROT	223
12.4.3	Testopdrachten	26
12.5	Tekst in grafiek	227
12.5.1	CHAR	227
12.5.2	TEXT	227
12.5.3	Testopdrachten	232
13.	**EEN KARAKTERSET MAKEN**	233
13.1	MEM	236
13.2	DESIGN 2	239
13.3	Apestaart	240
13.4	CSET 0/1	244
13.5	Testopdrachten	245
14.	**SPRITES (MOBs)**	247
14.1	Spritedefinities	250
14.1.1	DESIGN 0/1	251
14.1.2	Apestaart	252
14.1.3	MOB SET	254
14.1.4	CMOB	257
14.1.5	Testopdrachten	258
14.2	Spritebesturing	259
14.2.1	MMOB	259
14.2.2	MOB OFF	261
14.2.3	RLOCMOB	264
14.2.4	DETECT	267
14.2.5	CHECK	268
14.2.6	Testopdrachten	271
15.	**MUZIEK**	272
15.1	Hardwarevoorzieningen	272
15.2	VOL	274
15.3	WAVE	277
15.4	ENVELOPE	281
15.5	MUSIC	283
15.6	PLAY	285
15.7	Testopdrachten	287
16.	**STURENDE RANDAPPARATUUR**	289
16.1	Lightpen	289
16.1.1	PENX	290
16.1.2	PENY	291
16.2	Paddle en Joystick	292
16.2.1	POT	293
16.2.2	JOY	294

17. AANHANGSEL		295
17.1	Oplossingen van de testopdrachten	295
17.2	Kleurtabel	296
17.3	Foutmeldingen	296
17.4	Alfabetisch overzicht van de instructies	298
17.5	Simon's BASIC gegevens	299

1. INLEIDING

Direkt nadat we voor de eerste keer van Simon's BASIC hadden gehoord en we ons er wat intensiever mee bezig gehouden hadden, wisten we, dat dit een uitstekende vorm van uitbreiding is voor de Commodore 64.

U heeft zich laten overtuigen van de vele mogelijkheden van dit apparaat. Terwijl u thuis in uw CBM 64-gebruikershandboek zat te lezen, bent u misschien tot een vervelende ontdekking gekomen. Al die mogelijkheden, die zich onder het toetsenbord bevinden, zijn moeizaam en zeer langzaam onder de knie te krijgen en dat komt voornamelijk door het originele BASIC V2, de BASIC-versie die altijd aanwezig is in de computer. Grafieken, sound en sprites moeten door een grote hoeveelheid PEEKS en POKES, die bijna niet te doorgronden zijn en bovendien erg langzaam werken, worden gestuurd. Tegelijkertijd ontbreekt iedere mogelijkheid van een systematische en gemakkelijke programmering, zoals die wel in andere hogere programmeertalen is gerealiseerd. Zo staat voor het construeren van lussen slechts een instructie FOR...NEXT tot uw beschikking en dat geldt ook voor de vergelijkings- (IF) of sponginstructie (GOTO/GOSUB). De mogelijkheden voor het opstellen van een programma met LIST als enige opdracht, zijn echt niet zo bijzonder. Kortom: degene, die werkelijk serieus met zijn computer wil werken en niet alleen gebruik maakt van pasklaar-gemaakte spelen of andere gebruikersprogramma's zoals tekstverwerking of administratie van gegevens, heeft een BASIC-uitbreiding nodig.

Simon's BASIC richt zich tot de veelzijdige programmeurs. Het omvat opdrachten met betrekking tot (bijna) al die gebieden, die de Commodore 64 tot "Computer van het jaar 1983" hebben gemaakt. Natuurlijk zullen specialisten, die zich uitsluitend met een speciaal thema-gebied bezighouden, zoals grafiek of muziek, aangewezen zijn op een speciale uitbreiding of een andere programmavorm. Voor de beginners en voor de gevorderden in BASIC-programmering werkt Simon's BASIC echter optimaal. Omvangrijke programmeerhulp, zoals het definiëren van de functie-toetsen, list-hulpen of het opnieuw nummeren van BASIC-regels, bieden u de mogelijkheid tot een snelle en gemakkelijke opstelling van het programma. Waar u vroeger onnodig lange opdrachten moest intypen, is nu soms een druk op de toets voldoende.

Vele structurerende instructies, zoals REPEAT, symbolische adressering en dergelijke, die uit meer professionele programmeertalen stammen, vergemakkelijken het maken van een productiever programma. De opmaak van het beeldscherm en het vervaardigen van graphics met

alles wat daarbij hoort (sprites, verandering van tekens, multicolour en grafieken met een hoogoplossend vermogen), dat bieden u de mogelijkheden van de Commodore 64 met Simon's BASIC. Zo ondersteunen vele graphic-opdrachten het maken van veeleisende grafieken op het beeldscherm, die vanzelfsprekend ook met een printer kunnen worden afgedrukt.

Het geluid mag natuurlijk niet ontbreken, net zo min als hulp bij het uitlezen van de joystick, lightpen en paddle. Toch zijn er niet alleen maar positieve berichten te geven over Simon's BASIC. Het is duidelijk, dat geen enkel programma zonder fouten of gebreken is. Dat geldt ook voor Simon's BASIC. Het komt voor, dat verschillende functies niet juist worden uitgevoerd, of dat het programma weigert. We zullen u er natuurlijk bij helpen deze moeilijkheden te omzeilen. Sommige aanvullende opdrachten, waaronder een graphic-geheugen of meer diskette-opdrachten, die in het handboek slechts zeer onvolledig worden beschreven, kunt u ook in dit boek vinden.

In dit boek zullen we proberen voor u de afzonderlijke instructies zo begrijpelijk mogelijk, en vooral goed gerangschikt, weer te geven. Om telkens op de gebreken in het oorspronkelijke Simon's BASIC-Handboek te wijzen, leek ons, behalve bij zeer bezwaarlijke fouten, jammer van de energie. Daarom hebben we de opdrachten een geheel andere volgorde gegeven, die geschikt is voor een oefen- en leerboek waarin theorie en praktijk in een geleidelijk opklimmend niveau op elkaar zijn afgestemd. We beginnen bij de programmeerhulpmiddelen en de foutenbehandeling van gemakkelijke programmering, en we gaan verder met het grote hoofdstuk over de in- en outputcontrole, waarbij de graphics, sprite- en geluidsprogrammering zijn inbegrepen. Zoals u ziet, wordt er begonnen met opdrachten die voor de algemene programmering noodzakelijk zijn, om later op het gebruik van graphics en sound te zijn voorbereid. Als afsluiting van ieder hoofdstuk kunt u door een kleine test, waarvan de oplossingen aan het einde van het boek staan, uw kennis toetsen en zonodig vergroten.

We hopen u met deze beschrijving een leerzaam en informatief boek te hebben aangeboden. Speciale aandacht kunt u misschien besteden aan paragraaf 9.3, waarin 3 instructies worden beschreven die in het handboek niet worden genoemd en die tot op dit moment volledig onbekend waren.

2. PROGRAMMEERHULP

2.1 Het definiëren van de functietoetsen

De meeste Personal Computers bezitten functietoetsen voor het vereenvoudigd invoeren van vaak gebruikte opdrachten. Men maakt echter onderscheid tussen twee verschillende soorten functietoetsen:

- afzonderlijk definieerbare functietoetsen, die door de gebruiker in willekeurige volgorde van tekens of opdrachten kunnen worden voorzien.
- ASCII-functietoetsen, waaraan net als aan alle andere toetsen een ASCII-waarde is toegekend. De werkzaamheid van deze functietoetsen kan in het programma worden bepaald. Het verdere programma-verloop kan afhankelijk daarvan worden gestuurd.

De Commodore 64 is in de standaard-uitgave alleen uitgerust met ASCII-functietoetsen. Deze acht functietoetsen hebben de volgende ASCII-waarden:

F1 - 133 F3 - 134 F5 - 135 F7 - 136

F2 - 137 F4 - 138 F6 - 139 F8 - 140

In het programma kunt u de ASCII-waarde van deze toetsen met de volgende instructies opvragen:

```
100 GET X$:IF ASC(X$)< 133 OR ASC(X$)> 140 THEN 100
110 PRINT"DE ASCII-WAARDE VAN DEZE FUNCTIETOETS IS:";ASC(X)
120 GOTO 100
```

Dit programma accepteert alleen de functietoets-betekenis en toont dan de desbetreffende ASCII-waarde. Met de toets RUN/STOP kunt u dit programma verlaten. Dit gebruik is de eigenlijke zin en bedoeling van de functietoetsen. Met Simon's BASIC heeft u nu de mogelijkheid iedere functietoets met willekeurige opdrachten of tekens te definiëren. Dit is echter niet het enige, er werden bovendien acht

functietoetsen ingesteld, die met behulp van de Commodore-toets (C=) kunnen worden aangesproken. Het volgende overzicht geeft aan, hoe u die functietoetsen kunt aanspreken:

Functie-Toets	drukken
F1	F1
F2	SHIFT & F1
F3	F3
F4	SHIFT & F3
F5	F5
F6	SHIFT & F5
F7	F7
F8	SHIFT & F7
F9	C= & F1
F10	C= & SHIFT & F1
F11	C= & F3
F12	C= & SHIFT & F3
F13	C= & F5
F14	C= & SHIFT & F5
F15	C= & F7
F16	C= & SHIFT & F7

2.1.1. KEY

Format	KEY n,"string"
Parameter	n - nummer van de functietoets (1-16)
	string - reeks tekens of stringvariabele (max. 15 tekens)
Functie	definitie van een functietoets met een reeks tekens of instructies
Voorbeeld	KEY1,"LIST-100"
	Definieert de functietoets 1 met de instructie LIST

-100. Een RETURN wordt met de hand ingevoerd.

Met deze instructie kunt u nu uw eigen functietoetsen definiëren. Moet de functietoets tegelijkertijd de RETURN uitvoeren, dan moet aan de definitie de overeenkomstige ASCII-code voor RETURN worden toegevoegd. Probeer nu met behulp van de twee volgende voorbeelden het verschil vast te stellen:

```
KEY1,"PRINT 16*15"
KEY3,"PRINT 16*16"+CHR$(13)
```

Nadat u de twee functietoetsen hebt gedefinieerd, bedient u de functietoets F1 en daarna toets F3. U zult constateren, dat de functietoets F3 tegelijkertijd de noodzaak om de RETURN aan te slaan opheft. Dit is echter niet altijd zinvol. Instructies, waarvan een verkeerde bediening onaangename gevolgen kan hebben (bijvoorbeeld NEW) moeten zonder automatische RETURN in het geheugen worden opgeslagen. Wanneer u eens per ongeluk de functietoets voor NEW bedient, wordt de opdracht alleen geaccepteerd wanneer daarna RETURN wordt ingedrukt. U kunt de fout in de bediening dus nog tijdig vaststellen. Probeer nu eens de functietoetsen met instructies te definiëren. Gebruik de afsluitende toets RETURN. De functietoetsen kunnen ook in een programma worden vastgelegd, zodat de eigen definitie niet steeds weer met veel moeite hoeft te worden ingevoerd. Het volgende programma definieert de toetsen F1, F3, F5 en F7:

```
100 KEY1,"LIST"+CHR$(13)
110 KEY3,"GOTO"
120 KEY5,"PRINT"
130 KEY7,"OPEN1,8,15"
```

Wanneer deze vier functies steeds weer nodig zijn, kan dit programma in het geheugen worden opgeslagen. De bepaling van de functietoetsen met deze opdrachten is nu zeer eenvoudig. U laadt het programma van de functietoetsen en start dat door middel van de toets RUN. De gewenste functietoetsen staan u reeds ter beschikking.

De KEY-opdracht heeft een speciale eigenaardigheid: opdrachten met aanhalingstekens kunnen alleen op de functietoetsen worden vastgelegd, wanneer voor het aanhalingsteken een CHR$-code wordt gebruikt. Een kleine puzzel: probeer de functietoets F1 met de opdracht LOAD"$",8 te definiëren! Deze opdracht brengt de inhoudsopgave van de diskette in het geheugen. Voor een snelle oplossing kunt het volgende noteren:

```
KEY1,"LOAD"+CHR$(34)+"$"+CHR$(34)+",8"
```

Deze opdracht lijkt op het eerste gezicht onduidelijk, omdat het aanhalingsteken als CHR$(34) werd ingevoerd. U moet dus ieder gewenst aanhalingsteken in de tekenreeks vervangen door "+CHR$(34)+". Een andere mogelijkheid is de functietoetsen met behulp van een programma direkt te definiëren. De reeks tekens mag echter niet de tekens ":" en "," bevatten omdat deze door de INPUT niet worden geaccepteerd. Het volgende programma maakt het definiëren van de functietoetsen mogelijk in de vorm van een dialoog.

```
100 INPUT"NUMMER VAN FUNCTIETOETS: ";N
110 INPUT"DAARIN OPSLAAN: ";Z$
120 PRINT"RETURN BIJVOEGEN (J/N)?"
130 GET X$:IF X$<>"N" AND X$<>"J" THEN 130
140 IF X$="J" THEN Z$=Z$+CHR$(13)
150 IF LEN(X$)<16 THEN 170
160 PRINT"MAXIMAAL 15 TEKENS!":GOTO 180
170 KEY N;Z$
180 PRINT"NOG MEER OPSLAAN (J/N)?"
190 GET X$:IF X$<>"N" AND X$<>"J" THEN 170
200 IF X$="J" THEN 100
210 END
```

Dit programma vereenvoudigt het vastleggen van de functietoetsen na het inschakelen van de computer. Het accepteert het invoeren van het aanhalingsteken wel, maar de komma en de dubbele punt niet. Een RETURN wordt op verzoek toegevoegd. De lengte van de tekenreeks mag inclusief RETURN de 15 niet overschrijden. Een tip: Wanneer u de functietoets 1 met KEY1"?FRE(0)"+CHR$(13) definieert, krijgt u door een druk op de knop de opgave van het actuele vrije geheugen.

We moeten nog even een opmerking over fouten kwijt. De diskette die bij Simon's BASIC verkrijgbaar is, maakt het onmogelijk, een definitie van de functietoetsen ongedaan te maken. Voert u bijvoorbeeld de opdracht KEY1"" in, dan wordt de bestaande functie opgeheven en de nieuwe inhoud met ondefinieerbare tekens aangevuld! Laat u er niet door irriteren, want deze 'fout' is nauwelijks van betekenis. U kunt toch met een hernieuwde KEY-opdracht de functietoets naar believen veranderen.

2.1.2 DISPLAY

Format	DISPLAY
Parameter	geen
Functie	definitie van de funtietoetsen opgeven
Opmerking	in plaats van DISPLAY kan ook KEY 0 worden ingetoetst

Het definiëren van de functietoetsen is weliswaar een stuk gemakkelijker gemaakt, maar toch kan het overzicht hierover verloren gaan. Een situatie als "heb ik nu de LIST-opdracht op toets F1 of op toets F3 vastgelegd", is bij 16 definieerbare functietoetsen niet te voorkomen. Simon's BASIC biedt een vereenvoudiging van het gebruik van de vaak verwaarloosde functietoetsen, zodat u niet verscheidene functietoetsen hoeft te proberen om uw doel te bereiken. De DISPLAY-instructie geeft alle gedefinieerde posities van de functietoetsen aan op het beeldscherm. Aan de inhoud van de functietoetsen is de overeenkomstige KEY-instructie voorafgegaan. Een DISPLAY na het laden van Simon's BASIC geeft:

```
KEY1,""
KEY2,""
KEY3,""
KEY4,""
KEY5,""
KEY6,""
KEY7,""
KEY8,""
KEY9,""
KEY10,""
KEY11,""
KEY12,""
KEY13,""
KEY14,""
KEY15,""
KEY16,""
```

Bij de start van Simon's BASIC zijn dus alle functietoetsen "leeg". De DISPLAY-weergave op het beeldscherm kan natuurlijk samengesteld worden, dat wil zeggen: wilt u de een of andere functietoets opnieuw

definiëren, dan beweegt u na de DISPLAY de cursor naar de desbetreffende regel, verandert de KEY-instructie, en drukt op RETURN. Deze hulp bij de input gebruikt u ook bij het inschakelen en veranderen van programma's. Hier kan ook na de LIST de gewenste regel op deze manier worden veranderd. U bespaart zich daarmee dus de hernieuwde invoer van de regel.

Natuurlijk kunt u de door DISPLAY gegeven weergave ook op de printer zetten. Open eerst het kanaal van de printer met 'OPEN 1,4'. Leid daarna de volgende beeldschermweergave met 'CMD1' naar dit kanaal. De aansluitende 'DISPLAY' drukt de lijst met de definities van de functietoetsen af. Sluit nu het kanaal van de printer met CLOSE1.

2.1.3 Testopdrachten

1. Zoals bekend wordt de werking van de functietoetsen met KEY bepaald. Een van de volgende instructies is onjuist:

   A) KEY1,"LIST"
   B) KEY0,"RUN"
   C) KEY B,A$

2. Aan functietoets 5 moet het commando LIST toegevoegd worden. Automatisch moet dan RETURN in werking worden gesteld. Welke van de volgende instructies moet dan gegeven worden?

   A) KEY5,"LIST"+"RETURN"
   B) KEY5,"LIST"+CHR$(34)
   C) KEY5,"LIST"+CHR$(13)

3. Het commando LOAD"$",8 moet inclusief een automatische RETURN worden toegewezen aan functietoets 1. Welk commando doet dat?

   A) KEY1,"LOAD"$",8"+CHR$(13)
   B) KEY1,"LOAD"$",8"+CHR$(34)
   C) KEY1,"LOAD"+CHR$(34)+"$"+CHR$(34)+",8"+CHR$(13)
   D) KEY1,"LOAD"+CHR$(13)+"$"+CHR$(13)+",8"+CHR$(34)

4. Met hoeveel tekens kan een functietoets maximaal bezet worden?

A) 255
B) 15
C) 16
D) 256

2.2 Gemakkelijke programmasamenstelling

2.2.1 AUTO

Format	AUTO sn,sg
Parameter	sn - startnummer van regelnummering
	sg - stapgrootte van regelnummering (1-255)
Functie	automatische regelnummering met een variabel startnummer en variabele stapgrootte
Voorbeeld	AUTO 100,10
	De beginregel van het volgende programma is 100 en de stapgrootte wordt 10

Zoals alom bekend, worden de regels van het BASIC-programma van regelnummers voorzien, die door de interpreter op volgorde worden uitgevoerd. Het is weliswaar mogelijk met de stapgrootte 1 te nummeren, maar dan kunnen op een later tijdstip geen regels meer worden tussengevoegd. Daarom moet u een geschiktere stapgrootte kiezen. Meestal wordt met een stapgrootte van 10 gewerkt. Wanneer u de zelfstandige input van nummers voor iedere regel niet bevalt, is deze AUTO-opdracht op zijn plaats. Deze instructie geeft telkens het regelnummer, waaraan dan de inhoud wordt toegevoegd. Na het bedienen van RETURN, waarmee u aansluit bij de input van de nieuwe regel, wordt het volgende regelnummer gegeven enzovoort. Bedient u na het voorafgegeven regelnummer de toets RETURN, dan wordt de AUTO weer opgeheven en kan men normaal doorwerken. Let er echter

op, dat de AUTO-functie het einde van de regelnummering (63999) niet onderscheidt. Test dit met het volgende voorbeeld:

```
AUTO 63500,100
63500 PRINT
63600 PRINT
63700 PRINT
63800 PRINT
63900 PRINT
64000 PRINT


?SYNTAX ERROR
READY.
64100


?SYNTAX ERROR
READY.
```

Het bereiken van de nummeringsgrens wordt niet helemaal netjes opgevangen. De invoer van regelnummer 64000 veroorzaakt een ?SYNTAX ERROR en het regelnummer 64100 wordt aangegeven. Wilt u nu de AUTO-functie door RETURN verlaten, dan wordt eveneens eerst een ?SYNTAX ERROR gegeven. Maar omdat deze grens alleen in zeldzame gevallen wordt bereikt en dit ook geen ernstige gevolgen heeft, moet u dit maar voor kennisgeving aannemen. Ernstiger is het, wanneer u reeds regels hebt ingevoerd en een AUTO-opdracht neerzet, die deze regelnummers geeft. Wanneer u de AUTO-functie nu verlaat, is de gegeven regel weggeveegd. Een voorbeeld zal dit verduidelijken.

Veronderstel, dat u het volgende programma hebt ingevoerd:

```
100 FOR I=65 TO 90
110 PRINT"DE ASCII-WAARDE ";
120 PRINT I
130 PRINT " IS AAN HET TEKEN ";
140 PRINT CHR$(I)
150 PRINT "TOEGEWEZEN."
160 NEXT I
```

U hebt een list gemaakt van het programma en wilt nu andere regels vanaf 200 inschakelen. De desbetreffende opdracht zou 'AUTO 200,10' zijn. Nu maakt u een fout bij het invoeren en geeft de opdracht

'AUTO 100,10'. De computer reageert en geeft regel 100. Nu merkt u de input-fout op en u verlaat de functie met RETURN. Regel 100 is nu gewist.

De enige uitweg is de 'nooduitgang', dus de gelijktijdige bediening van de toetsen RUN/STOP en RESTORE, wanneer regel 100 wordt uitgevoerd. Hebt u het probleem herkend? Zo ja, let dan altijd op een juiste invoer van de AUTO-opdracht, wanneer er zich al een programma in het geheugen bevindt.

2.2.2 RENUMBER

Format	RENUMBER sn,sg
Parameter	sn - startnummer nieuwe nummering
	sg - stapgrootte van nieuwe nummering
Functie	opnieuw nummeren van de regels van een programma met een variabele stapgrootte van regelnummers onderling
Voorbeeld	RENUMBER 100,10
	De in het geheugen aanwezige programmaregels worden in stappen van 10 vanaf regel 100 doorgenummerd
Opmerking	sprongadressen van GOTO en GOSUB worden bij de nieuwe nummering niet aangepast!

Wanneer de tot nu toe gebruikte nummering van de BASIC-regels onoverzichtelijk lijkt of op veel plaatsen door het invoeren van nieuwe regels "een rommeltje" is geworden, helpt deze opdracht u verder. Hij nummert het hele programma opnieuw, waarbij de beginregel en de stapgrootte vrij te kiezen zijn. "Fantastisch" zullen velen van u denken, maar de mogelijkheden, die u met deze opdracht hebt, worden al gauw beperkt. Wat hebt u aan een compleet herziene nummering, als er geen rekening wordt gehouden met de sprongadressen van de GOTO- en GOSUB-opdrachten, en deze dus niet veranderd kunnen worden? De ontbrekende structureringsmogelijkheden van het

standaard-BASIC maken het onmogelijk zonder GOTO- en GOSUB te werken. Wanneer u echter de mogelijkheden van Simon's BASIC op de goede manier benut, zult u deze opdracht toch effectief weten in te zetten. Met Simon's BASIC is het mogelijk symbolische adressen toe te kennen. Zo kan een gewone opdracht als 'GOTO 100' in 'GOTO UITVOER' worden veranderd. Het toekennen van symbolische adressen bij alle GOTO- en GOSUB-opdrachten is dus niet te vermijden, als u RENUMBER gebruikt. Meer informatie over het toekennen van symbolische sprongadressen vindt u in paragraaf 5.2. Probeer nu deze instructie eens met behulp van het volgende programma:

```
10 PRINT"10"
11 PRINT"11"
12 PRINT"12"
15 PRINT"15"
20 PRINT"20"
```

De nummering is onregelmatig en tussen de regels 10 en 11 of 11 en 12 kan niets meer worden ingevoegd. In zo'n situatie is het zinvol om het programma opnieuw te nummeren. Geef daartoe de opdracht 'RENUMBER 10,10'. Wanneer u daarna het programma met 'LIST' op het beeldscherm brengt, ziet de nummering er als volgt uit:

```
10 PRINT"10"
20 PRINT"11"
30 PRINT"12"
40 PRINT"15"
50 PRINT"20"
```

Nu is de nummering weer correct en kunt u op alle plaatsen regels toevoegen. Doordat wij bij het maken van dit boek iedere opdracht zorgvuldig hebben gecontroleerd, stuitten we ook hier op een eigenaardigheid: een stapgrootte van 0 heeft als gevolg, dat alle regels van hetzelfde nummer worden voorzien. Een zo gemanipuleerd programma kan zelfs worden afgedraaid. Om u hiervan te overtuigen, kan na het invoeren van het programma aan het laatste voorbeeld de opdracht 'RENUMBER 10,0' worden toegevoegd. De aansluitende opdracht 'LIST' zal een ongewone LISTING op het beeldscherm brengen:

```
10 PRINT"10"
10 PRINT"11"
10 PRINT"12"
10 PRINT"15"
10 PRINT"20"
```

Door met 'RUN' te starten kunt u zich van de uitvoerbaarheid van het programma overtuigen. Wanneer u deze regels nu afzonderlijk wilt weghalen, voert u telkens '10' met de afsluitende RETURN in. 10 wordt dan gewist. Maar welke? De eerste regel 10 (PRINT"10") wordt gewist. Wanneer u nu nog eens regel 10 wist, wordt de regel 'PRINT"11"' gewist, enzovoort. Op deze manier kunnen programma's tegen ingrepen worden beschermd, omdat er geen regels meer kunnen worden veranderd of toegevoegd. Wilt u bijvoorbeeld de regel 'PRINT"15"' veranderen in 'PRINT"17"', dan wordt niet deze regel veranderd, maar regel 'PRINT"10"' wordt geruild voor regel 'PRINT"17"'. Het programma kan dus niet meer worden geredigeerd.

Een andere bijzonderheid:

Voert u een RENUMBER-opdracht in die ertoe leidt dat het maximale regelnummer 63999 wordt overschreden, dan kunnen deze regels noch worden aangesprongen, noch gewist. Het bewijs: nummer het zojuist beschreven programma met 'RENUMBER 63800,100'. Na LIST komen de volgende regels op het beeldscherm:

```
63800 PRINT"10"
63900 PRINT"11"
64000 PRINT"12"
64100 PRINT"15"
64200 PRINT"20"
```

De toegestane regelnummering mag echter de 63999 niet overschrijden. Hier wordt weer tegen alle regels in gehandeld. Het programma is wel te draaien, maar toch kunnen de regels vanaf 64000 niet meer gewist worden! Probeer het eens met de input '64000' en aansluitend RETURN. De melding '?SYNTAX ERROR' verschijnt. Ook kunnen de regels vanaf 64000 niet met GOTO of GOSUB worden gebruikt. Probeer het programma te draaien met

```
10 GOTO 64100
```

De interpreter meldt weer:

```
?SYNTAX ERROR
```

Alleen een correcte RENUMBER kan dit programma nog in orde maken.

2.2.3 MERGE

Format	MERGE "naam",an
Parameter	naam - naam van het programma dat geladen moet worden
	an - apparaatnummer waar vanaf geladen moet worden
Functie	bijladen van een extern opgeslagen programma
Voorbeeld	MERGE "INV-ROUT",8
	Het programma "INV-ROUT" zal vanaf de disk drive worden geladen achter het in het geheugen aanwezige programma dat daarbij behouden blijft.

Wie veel in BASIC programmeert, ontwikkelt individuele routines (subprogramma's), die in diverse programma's kunnen worden toegepast. Maar deze routines moeten wel gemakkelijk toegankelijk zijn als u ze veel wilt gebruiken. Een mogelijkheid daarvoor is de routine uit te printen en die uitdraai iedere keer weer over te tikken. Maar dat is te omslachtig. Het is zinvoller de routines op diskette op te slaan en met het hoofdprogramma te verbinden. Dit is echter gemakkelijker gezegd dan gedaan. Het bedrijfssysteem van de Commodore 64 biedt geen mogelijkheid om programma's zonder moeizame POKEs te verbinden.

Het MERGE-commando biedt nu de mogelijkheid, om vaak benodigde programmadelen op diskette te zetten en naar behoefte toe te voegen aan het programma, dat zich in het geheugen bevindt. U moet daarbij op drie dingen letten:

- 1. De regelnummers van het programma, dat zich in het geheugen bevindt en van het later toegevoegde programma mogen niet identiek zijn.
- 2. De regelnummers van het toegevoegde programma moeten hoger zijn dan die van het programma, dat zich in het geheugen bevindt.
- 3. Het vastleggen van de variabelen moet verschillend zijn. Dezelfde variabelen in verschillende programmadelen kunnen later

de oorzaak zijn van vele moeilijk op te lossen fouten.

De beste manier om problemen bij het MERGEn van programma's te vermijden, is een duidelijke, uniforme structurering van ALLE programma's. Wijs aan ieder vaker gebruikte subroutine een (voor altijd vastgelegd) regelnummergebied toe, waarin u verder in principe niet programmeert. U kunt zo ook het beste bij de variabelen te werk gaan. U zult zich afvragen, welke effecten het heeft, wanneer u deze 3 regels bij de MERGE-instructie niet in acht neemt. Doordat experimenteren meer uitwerking heeft dan studeren, passen we in het volgende voorbeeld de MERGE-instructie zonder meer toe.

Voer eerst het programma in, dat later moet worden toegevoegd:

```
10 PRINT "REGEL 10, HOOFDPROGRAMMA"
20 PRINT "REGEL 20, HOOFDPROGRAMMA"
30 PRINT "REGEL 30, HOOFDPROGRAMMA"
```

Bewaar dit later toegevoegde programma op schijf door middel van 'SAVE "TEST.MERGE",8' of op cassette door middel van 'SAVE"TEST"'. Wis nu dit programma, dat zich in het geheugen bevindt, door het commando 'NEW'. Omdat we het zojuist weggeschreven programma aan een reeds aanwezig programma willen toevoegen, tikken we een programma in, maar doen dit in dit geval met dezelfde regelnummers:

```
10 PRINT "REGEL 10, NAGELADEN"
20 PRINT "REGEL 20, NAGELADEN"
30 PRINT "REGEL 30, NAGELADEN"
```

Nu zullen we eens kijken, hoe Simon's BASIC reageert, wanneer het programma, dat net in het achtergrondgeheugen werd opgeslagen en dat toch dezelfde regelnummers bevat, wordt nageladen. Voer daarvoor het volgende commando in.

```
MERGE "TEST.MERGE",8
```

of

```
MERGE "TEST.MERGE"
```

Het apparaat-adres 8 moet alleen worden aangegeven, wanneer u vanaf de diskdrive laadt. Nadat de computer met 'READY' terugkomt, toont

de opdracht 'LIST' wat er zojuist is gemaakt.

```
10 PRINT "REGEL 10, HOOFDPROGRAMMA"
20 PRINT "REGEL 20, HOOFDPROGRAMMA"
30 PRINT "REGEL 30, HOOFDPROGRAMMA"
10 PRINT "REGEL 10, NAGELADEN"
20 PRINT "REGEL 20, NAGELADEN"
30 PRINT "REGEL 30, NAGELADEN"
```

Dit "programma" is een vreemde ervaring voor iedereen, die al langer dan twee dagen programmeert. Geen van de regels voor de opbouw van een BASIC-programma zijn in acht genomen; iedere regel is tweemaal aanwezig. Laten we nu even controleren, of de BASIC-interpreter deze wanorde verdraagt. Start daarvoor het programma met 'RUN'. Het succes is verbazingwekkend: De interpreter zwaait niet met de witte vlag, maar werkt zonder protest het programma af.

Om dit te verklaren, moet de uitleg iets uitvoeriger zijn. Na de opdracht 'RUN' wordt eerst de positie (het adres) van de eerste BASIC-regel opgespoord. Dit adres bevindt zich op de zogenaamde zero-page, de eerste 256 bytes van het hoofdgeheugen. Hier wordt informatie opgeslagen, die het bedrijfssysteem of de BASIC-interpreter in de routines nodig heeft. Dit is vergelijkbaar met een BASIC-programma, dat belangrijke informatie in de variabelen afzet. De zero-page is dus het 'variabelen-gebied' van het bedrijfssysteem en de interpreter. Het adres van de eerste BASIC-regel (bij CBM 64 altijd $0801 of decimaal 2049) wordt dus eerst vastgesteld en op de zero-page als "actuele regel" vastgelegd. Nadat de regel nu is afgewerkt, wordt het adres van de volgende regel als "actuele regel" opgeslagen. Dit adres van de volgende regel bevindt zich aan het begin van iedere BASIC-regel en wordt bij de opdracht 'LIST' niet aangegeven. De verbinding van regel naar regel wordt dus niet door het eigenlijke regelnummer, maar door een speciaal vervolgadres gemaakt. Het regelnummer is dan bij het afdraaien van het programma alleen bij spronginstructies (GOTO, GOSUB) nodig. Hier wordt de aangeslagen regel gezocht en benoemd tot "actuele regel". Het einde van het programma wordt herkend aan het vervolgadres 0. Een programma met diverse gelijke regelnummers wordt dus pas bij sprongcommando's beëindigd. Omdat iedere regel dubbel aanwezig is, is het nu interessant om vast te stellen, hoe op een 'GOTO 10' wordt gereageerd. Wordt de eerste regel 10 aangesproken, de tweede of...? Kijk zelf maar: voer de opdracht 'GOTO 10' in en u zult constateren, dat steeds de eerste regel wordt aangesproken. Dit kan verklaard worden door het feit, dat de interpreter bij het zoeken naar de aangeslagen regel steeds bij het begin van het programma begint. Een zo verwarrend programma gebruikt u natuurlijk niet. Gebruik het

RENUMBER-commando om hier orde op zaken te stellen: 'RENUMBER 10,10'. Het resultaat, dat U met 'LIST' kunt bekijken, ziet er nu als volgt uit:

```
10 PRINT "REGEL 10, HOOFDPROGRAMMA"
20 PRINT "REGEL 20, HOOFDPROGRAMMA"
30 PRINT "REGEL 30, HOOFDPROGRAMMA"
40 PRINT "REGEL 10, NAGELADEN"
50 PRINT "REGEL 20, NAGELADEN"
60 PRINT "REGEL 30, NAGELADEN"
```

Let op:
Op de zero-page wordt niet alleen het begin van het BASIC-programma in het geheugen opgeslagen, maar ook het einde. Wanneer het BASIC-einde in een programma met POKE-commando's wordt opgezet, kan een MERGE problemen opleveren. Voor programmeurs, die dergelijke experimenten niet uitvoeren, geldt dit natuurlijk niet. Mocht u echter toevallig deze wijzer van het BASIC-einde manipuleren, zodat een MERGE geen effect heeft, dan kan met de commando's 'NEW' en 'OLD' het BASIC-einde weer worden aangepast.

2.2.4 Testopdrachten

1. De automatische regelnummering moet worden ingeschakeld met beginregel 100 en stapgrootte 10. Welk commando moet hiervoor worden gegeven?

    A) AUTO 100,10
    B) AUTO 10,100
    C) AUTO 100 BY 10
    D) AUTO 10 BY 100

2. De stapgrootte van de AUTO-opdracht is beperkt tot:

    A) 255
    B) 256
    C) onbeperkt.

3. Welke opdrachten worden door het RENUMBER-commando niet naar behoren uitgevoerd?

   A) FOR/NEXT - lussen
   B) GOTO
   C) GOSUB
   D) INPUT

4. Waarop moet u letten bij het regelmatig gebruik van het MERGE-commando?

   A) De regelnummers moeten hoger zijn dan 100
   B) Er mogen geen identieke regelnummers voorkomen.
   C) De regelnummers van het later ingevoerde programma moeten hoger zijn dan de laatste regel van het bestaande programma.
   D) Voor het gebruik van MERGE moet het programma door de opdracht RENUMBER opnieuw worden doorgenummerd.

2.3 LIST-hulpmiddelen

Voor het bewerken van een BASIC-programma is inzicht in de listing noodzakelijk. Het meest overzichtelijk is een uitgeprinte BASIC-listing. Niet iedereen beschikt echter over een printer. Een printuitdraai is ook niet altijd zinvol. Wanneer u bijvoorbeeld eens iets wilt veranderen, is het erg tijdrovend om het hele programma uit te printen. Het BASIC-programma moet dus ook overzichtelijk op het beeldscherm verschijnen. Bij de uitvoer van BASIC-listings op het beeldscherm is de CBM 64 nogal ongemakkelijk. U kunt alleen met het commando 'LIST' een lijst van het programma weergeven, de LIST-snelheid met de toets 'CTRL' regelen en op de betrokken plaats met de toets 'RUN/STOP' afbreken. Wanneer u nu vanaf het afgebroken regelnummer verder wilt listen, is weer een 'LIST'-commando nodig. In het standaard-BASIC zijn de volgende mogelijkheden aanwezig voor het gebruik van de 'LIST'-instructie.

LIST	het hele programma wordt getoond
LIST n	een bepaalde regel wordt getoond (LIST 100)
LIST -n	het programma wordt tot regel n getoond (LIST -500)
LIST n-	het programma wordt vanaf regel n getoond (LIST 100-)

LIST n1-n2	het programma wordt van regel n1 tot regel n2 getoond (LIST 100-500)

Met Simon's BASIC hebt u andere mogelijkheden, die de weergave van uw programma sturen.

2.3.1 PAGE

Format	PAGE n
Parameter	n - aantal regels per pagina minus 1 (1-23)
Functie	indelen van programmalistings in pagina's
Voorbeeld	PAGE 10
	Bij de volgende uitvoering van LIST wordt het programma in gedeeltes van 10 regels getoond.
Opmerking	De instructie PAGE 0 schakelt de PAGE-functie weer uit.

Met behulp van deze instructie kunt u nu het BASIC-programma in pagina's indelen. De grootte van de pagina's kan vrij worden gekozen, maar een pagina mag niet groter zijn dan een beeldscherminhoud. U moet erop letten, dat het bij het aangeven van regels gaat om de beeldschermregels en niet om de programmaregels. Bij deze opdracht hebben we het volgende op te merken: kiest u een waarde hoger dan 23, dan wordt geen PAGE-functie in werking gesteld. Het zou hier zinvoller zijn, wanneer de foutmelding, die bij een waarde van n groter dan 255 wordt aangegeven (ILLEGAL QUANTITY ERROR) ook bij een waarde van n groter dan 23 actief wordt; deze waarde heeft immers geen effect. De procedure voor het gebruik van dit commando is als volgt.

- 1. U geeft een PAGE-instructie met de gewenste paginagrootte op.
- 2. U start de uitvoering met LIST.
- 3. U bladert het programma met de RETURN door.

- 4. U onderbreekt de uitvoering naar behoefte met de toets RUN/STOP.

De onderbreking van de uitvoer is nog niet zo eenvoudig als het lijkt. U kunt weliswaar de LIST-output met de toets RUN/STOP onderbreken, maar alleen tijdens de OUTPUT, niet wanneer de volledige regels van de paragraaf al op het beeldscherm staan en de computer wacht op RETURN voor de output van de volgende paragraaf. Probeer dit eens uit. Laad een van uw programma's en voer aansluitend de instructie 'PAGE 10' in. Wanneer u nu met 'LIST' de output van de eerste pagina inleidt, verschijnt eerst het eerste regelnummer van het programma op het beeldscherm. Drukt u nu op de toets RETURN, dan wordt het beeldscherm gewist en het eerste deel van het programma wordt weergegeven. Nu kunt u de output echter niet onderbreken. U moet daarvoor nog eens de RETURN-toets indrukken en tijdens het lopen van de volgende output de toets RUN/STOP bedienen.

Met de PAGE-functie kan natuurlijk ook elke variatie van het LIST-commando worden ingevoerd, die al eerder werden beschreven.

2.3.2 OPTION

Format	OPTION n
Parameter	n - 10 voor inschakelen; ongelijk 10, maar kleiner dan 256 voor uitschakelen
Functie	in de listing van programma's worden alle instructies van Simon's BASIC in reverse voorgesteld en dus benadrukt.
Voorbeeld	OPTION 10 - schakelt de onderscheiding in
	OPTION 0 - schakelt de onderscheiding uit

Het op de voorgrond plaatsen van alle Simon's BASIC-opdrachten kan erg zinvol zijn. Stel u eens voor, dat een bepaald programma, geschreven met behulp van Simon's BASIC, in standaard-BASIC moet worden omgezet. Het is erg tijdrovend het hele programma naar deze opdrachten af te zoeken. Wanneer u echter deze instructies door middel van 'OPTION 10' accentueert, zal het opsporen daarvan

gemakkelijker zijn.

Een voorbeeld: laad een programma, dat met Simon's BASIC-opdrachten is uitgerust. Misschien hebt u het programma uit paragraaf 2.1.1 als definiëring van de functietoetsen ingevoerd. Wanneer u nu 'OPTION 10' en daarna 'LIST' invoert, ziet u het effect van de OPTION-opdracht: alle instructies van Simon's BASIC zijn reverse weergegeven. Deze opdrachten worden op sommige printers, die een Commodore-Modus hebben, eveneens reverse weergegeven.

2.3.3 DELAY

Format	DELAY n
Parameter	n - vertragingswaarde (1-255)
Functie	verandert de LIST-snelheid met instelbare vertragingswaarde
Voorbeeld	DELAY 10 De volgende uitvoering van LIST wordt met behulp van de SHIFT-toets tot ongeveer 50 tekens per seconde vertraagd

Naast de standaard-vertraging van de listings door middel van de CTRL-toets, kan nu met de toets SHIFT een variabele vertraging worden gestuurd. De vertragingsfactor kan door de DELAY-opdracht worden ingesteld. Wanneer u de Commodore-toets (C=) ingedrukt houdt, wordt de LIST-output gestopt.

Hier volgt een overzicht, zodat u een idee van deze vertragingswaarde krijgt:

Vertragingswaarde	Teken/sec.
1	290
5	100
10	50
20	25

De waarde 1 komt ongeveer overeen met de normale LISTsnelheid en de waarde 10 met de vertraging door de CTRL-toets. De PAGE- en de DELAY-functie kunnen met elkaar worden gecombineerd. Dat wil zeggen, dat de listing in delen kan worden opgesplitst, waarvan u de weergave door de SHIFT-toets kunt vertragen. U hebt waarschijnlijk al snel uw persoonlijke PAGE- en DELAY-waarde ontdekt, die u dan bijvoorbeeld bij elk programma als eerste regel voorop kunt zetten. Deze regel kan er bijvoorbeeld zo uitzien:

```
1 PAGE 10:DELAY 5
```

Wanneer u het programma na het laden start, staat na een onderbreking uw persoonlijk opgespoorde LISThulp tot uw beschikking.

2.3.4 FIND

Format	FIND code of FIND string
Parameter	code - alles wat zich buiten aanhalingstekens bevindt
	string - alles wat zich binnen aanhalingstekens bevindt
Functie	het hele programma wordt op een bepaalde reeks tekens doorzocht.
Voorbeeld	FINDPRINT Geeft alle regels op waar de instructie PRINT in voorkomt
Opmerking	achter de FIND-instructie mag slechts een spatie voorkomen als die ook inderdaad wordt gezocht.

Deze instructie zullen heel weinig Simon's BASIC-gebruikers ooit effectief hebben toegepast. Reden daarvoor is, dat de instructie in het erbij geleverde handboek in 11 regels wordt beschreven; voor een volledige beschrijving zijn echter wel 11 pagina's nodig! Wat kunt u nu met deze instructie doen? Voordat alles nauwkeurig wordt beschreven, moet u letten op de volgende belangrijke bijzonderheid van deze instructie. U weet, dat in BASIC spaties tussen instructie en parameter geen effect hebben. Zo hebben bijvoorbeeld de instructies

'PRINT 10' en 'PRINT10' dezelfde uitwerking. De FIND-instructie vormt hierop een uitzondering: als u tussen instructie en parameter een spatie invoert, wordt deze spatie ook gezocht. Een voorbeeld: 'FIND10' zou bijvoorbeeld een regel met de code 'PRINT10' aangeven, dit gebeurt niet met 'FIND 10'. Zoals u al uit de beschrijving van de parameters zult hebben begrepen, zijn er twee verschillende manieren om te zoeken. Het zoeken naar code en het zoeken naar strings. Om het verschil te zien, moet u het volgende voorbeeld-programma invoeren.

```
10 PRINT"VOER EEN WAARDE IN VOOR A"
20 INPUT A
30 PRINT"U HEEFT DE WAARDE";A;"INGEVOERD"
```

Alle regels, die een 'A' bevatten, moeten worden getoond. Dus

```
FINDA
```

En nu de hamvraag: welke regels worden getoond? Opdat u niet te lang hoeft na te denken, volgt hier het antwoord:

De regels 20 en 30 worden aangegeven.

U kunt nu natuurlijk verder vragen naar het waarom, maar het gaat er nu niet om instructies op zich te onderzoeken. Dat hebben wij al voor u gedaan. Regel 10 wordt niet aangegeven, omdat hierin weliswaar een 'A' aanwezig is, maar deze staat in een string (tussen aanhalingstekens). Laten we nu verder experimenteren. Geef de instructie

```
FIND A
```

en u zult zien, dat alleen regel 20 wordt weergegeven, omdat dit de enige regel is die in de code een spatie bevat gevolgd door een 'A'. De 'A' in regel 30 heeft geen voorafgaande spatie. Een ander voorbeeld: De instructie 'FINDPRINT' resulteert in de weergave van de regels 10 en 30, omdat hierin een 'PRINT' aanwezig is. De instructie 'FIND PRINT' geeft echter geen enkele regel weer, omdat de spatie tussen regelnummer en regelinhoud door de BASIC-interpreter zelfstandig wordt neergezet; het is dus geen echte spatie. Natuurlijk kunt u voor het opsporen van BASIC-instructies ook afkortingen gebruiken. Zo is bijvoorbeeld 'FINDPRINT' identiek aan 'FIND?'.

Nu kan natuurlijk ook ieder willekeurig extra-teken worden opgezocht. De instructie 'FIND;' bijvoorbeeld zoekt naar alle puntkomma's. Het zoeken naar strings is niet zo veelzijdig als het zoeken naar codes. Wilt u bijvoorbeeld nagaan in welke regel 'VOER DE WAARDE A IN' aanwezig is, dan is daarvoor de instructie 'FIND"VOER DE WAARDE A IN"' nodig. U moet de gezochte string dus tussen aanhalingstekens zetten. De instructie FIND biedt een vereenvoudiging bij het zoeken naar strings. U kunt na een bepaald begin een string zoeken. Zo geeft bijvoorbeeld de instructie 'FIND"G' alle regels met strings die met 'G' beginnen. U kunt echter geen onderdelen in het midden of aan het einde van een string opzoeken. Voer het volgende programma in om met de instructie 'FIND' vertrouwd te raken en let op de aansluitende tabel met verschillende FIND-mogelijkheden:

```
10 REM VOORBEELD BIJ DE FIND-INSTRUCTIE
20 PRINT"VOER EEN WAARDE IN VOOR A"
30 INPUT A
40 PRINT"VOER EEN WAARDE IN VOOR B"
50 INPUT B
60 C=A+B
70 PRINT"DE SOM VAN DE TWEE WAARDEN"
80 PRINT"IS";C
```

FIND-INSTRUCTIE	GEVONDEN REGELS
FINDREM	10
FINDFIND	10
FINDA	10 30 60
FIND A	30
FINDB	10
FIND B	10 50
FIND"VOER DE	20 40
FINDC	10 60 80
FINDC=	60
FINDA+B	60
FINDPRINT	20 40 70 80
FINDINPUT	30 50
FIND;	20 40 80

2.3.5 Testopdrachten

1. Welke van de volgende instructies splitst de programmalisting op in delen van 10 beeldschermregels ?

   A) PAGE 10
   B) PAGE 11
   C) PAGE 9

2. Met welke toets kunt u de tekst-delen van de door PAGE opgedeelde programmalisting "doorbladeren" ?

   A) met RETURN
   B) met de Commodore-Toets (C=)
   C) met SHIFT
   D) met CTRL

3. Welk commando schakelt de markering van Simon's BASIC-instructies in?

   A) OPTION
   B) OPTION 1
   C) OPTION 10
   D) OPTION 0

4. Welke instructie verandert de snelheid van de programmalisting?

   A) DECAY
   B) DELAY
   C) DELLAY

5. Welke toets vertraagt de snelheid tijdens het listen van het programma op het beeldscherm?

   A) RETURN
   B) SHIFT
   C) Commodore-Toets (C=)

6. Het volgende programma is gegeven:

```
10 INPUT "VOER EEN GETAL IN";A
20 PRINT "U HEBT";A;"INGEVOERD
```

Welke van de volgende FIND-instructies geeft geen regel als gevonden weer?

A) FINDA
B) FIND A
C) FIND"G

3 DE BEHANDELING VAN FOUTEN

Zelfs de beste programmeurs kunnen zich niet onttrekken aan een intensieve toetsing en de daaruit resulterende foutenanalyse van hun programma's. Welke mogelijkheden biedt de CBM 64 in de standaarduitvoering voor de behandeling van fouten? We onderscheiden twee verschillende manieren om fouten op te sporen: de externe en de interne maatregelen. Fouten noemen we in het computerjargon BUGS en het opsporen van fouten DEBUGGING. Externe maatregelen zijn bijvoorbeeld de weergave van variabelen na een programma-onderbreking of een gerichte LIST op de door de interpreter als fout aangegeven regel. Interne maatregelen zijn opdrachten, die weliswaar in het programma zijn ingebouwd, maar niet bij het eigenlijke programma horen. Zij worden na een afgesloten foutentest weer verwijderd of als REMARKS (REM) buiten werking gesteld.

De Commodore 64 bevat een instructie, die uitsluitend voor de interne foutmelding is bedoeld. Dit is het commando STOP, waarvan functie en mogelijkheden bij slechts weinig mensen bekend zijn. Een STOP wordt in dit programma op een plaats ingebouwd, waar een onderbreking van het programma voor aansluitende controle van variabelen nodig is. De computer meldt bij het bereiken van een STOP: 'BREAK IN...'. Daarna kunnen alle gewenste variabelen worden weergegeven (bijvoorbeeld met PRINT A$). Wanneer het programma moet worden voortgezet, voert u het commando CONT in. Het programma loopt nu door tot het einde of tot de volgende STOP.

BELANGRIJK:

Wordt er na een programma-onderbreking een ingreep in het programma ondernomen (regels veranderd bijvoorbeeld), dan is door CONT geen voortzetting van het programma meer mogelijk. Ook alle variabelen worden gewist. De computer meldt: "?CAN'T CONTINUE ERROR". Variabelen kunnen echter na een programma-onderbreking naar behoefte worden bewerkt. Zo kan aan een variabele een testwaarde worden toegekend en het programma kan met CONT worden voortgezet. Andere commando's, die speciaal voor de foutenbehandeling gelden, heeft de standaard Commodore 64 niet. Alle andere maatregelen voor het zoeken van fouten moet de programmeur zelf nemen. Hij kan bijvoorbeeld bij het inlezen van een bestand in de computer met een ingevoegde PRINT alle gegevens weergeven, die de computer opneemt.

Deze hulpmiddelen, die in het programma zijn geïntegreerd voor het opsporen van fouten, moeten na succesvolle correctie weer worden

verwijderd. Hoe denkt u, dat een gebruiker van uw programma zal reageren, wanneer - bijvoorbeeld bij de weergave door de computer - de melding 'BREAK IN 1220' verschijnt? Het is zinvol, om de regels te noteren waarin u interne maatregelen inbouwt voor het opsporen van fouten. Dan is het aan het einde van de programmatoetsing gemakkelijk om alle regels te normaliseren. Zoals reeds gezegd, bezit de Commodore 64 alleen de commando's STOP en CONT als ondersteuning bij de foutenopsporing. Simon's BASIC biedt ook hier enkele aanvullende commando's.

3.1 Externe foutenbehandeling

3.1.1 TRACE

Format	TRACE n
Parameter	n - 10 voor inschakelen en ongelijk 10 voor uitschakelen
Functie	Opgave van alle net doorlopen programmaregels op het beeldscherm
Voorbeeld	TRACE 10
	schakelt de TRACE-modus in.

TRACE is een geliefde functie voor de foutenbehandeling. In comfortabele BASIC-interpreters, zoals MBASIC van MICROSOFT, is zo'n functie aanwezig. Nu hoeven ook de gebruikers van de Commodore 64 niet van TRACE af te zien. De werkwijze van dit commando is gemakkelijk uit te leggen. Schakel de TRACE-modus in, voordat u het programma met "TRACE 10' start. Wanneer u nu het programma start, verschijnt in de rechter bovenhoek van het beeldscherm een venster van 6 regels met ieder 6 tekens. In dit venster wordt altijd de laatst doorlopen regel weergegeven. Natuurlijk wordt het programma daardoor wat trager, omdat de computer naast uw programma ook het TRACE-"programma" moet sturen. Het volgende programma-voorbeeld zal de TRACE-functie verduidelijken:

```
5 PRINT CHR$(147);
10 PRINT CHR$(19);"10":GETX$:IFX$=""THEN10
20 PRINT CHR$(19);"20":GETX$:IFX$=""THEN20
30 PRINT CHR$(19);"30":GETX$:IFX$=""THEN30
40 PRINT CHR$(19);"40":GETX$:IFX$=""THEN40
50 PRINT CHR$(19);"50":GETX$:IFX$=""THEN50
60 END
```

De speciale tekens voor 'CLR/HOME' en 'HOME' zijn door CHR$(147) en CHR$(19) gecodeerd, omdat de speciale tekens niet aan iedereen bekend zijn. Voer dus dit programma in. Nu komt het grote ogenblik: schakel de TRACE-modus in met 'TRACE 10' en start het programma met 'RUN'. De laatste regel van het beeldschermvenster geeft de laatst-verwerkte regel weer (regel 5). De PRINT in regel 10 is al uitgevoerd, maar regel 10 is nog niet volledig uitgewerkt. U moet dus nog een toets indrukken. Druk op een willekurige toets, zodat regel 10 in het TRACE-venster verschijnt, en de PRINT in regel 20 toont dat het programma zich juist in regel 20 bevindt. Zo gaat het verder, tot het programma is beëindigd. Aan het einde van het programma vindt u het volgende TRACE-venster:

```
#5
#10
#20
#30
#40
#50
```

Regel 60 is weliswaar ook uitgevoerd, maar wordt niet meer weergegeven, omdat het einde van het programma is bereikt. Schakel nu de TRACE-modus weer uit met 'TRACE 0'. Controleer door middel van 'RUN', of TRACE inderdaad is uitgeschakeld.

Deze demonstratie heeft de werkwijze van het TRACE-commando verduidelijkt. Maar ook bij dit commando moet u opletten: schakel de TRACE-modus nog eens in met 'TRACE 10' en start het programma met 'RUN'. Onderbreek de loop van het programma met de toets 'RUN/STOP'. De computer meldt nu 'BREAKIN 10'. Wanneer u nu in de actuele vijfde regel de opdracht 'LIST' invoert, verschijnt de melding '?SYNTAX ERROR'. Nu zult u zich misschien afvragen, waarom de computer de opdracht niet accepteert. Het antwoord is echter simpel: in regel 5, waarin het LIST-commando is ingevoerd, bevindt zich aan het eind van de regel de rand van het TRACE-venster. Deze rand wordt mede als commando geïnterpreteerd en dit heeft de foutmelding als gevolg. U moet nu of het beeldscherm wissen, of de cursor buiten de eerste 6 regels bewegen, voordat u verdere commando's invoert.

Omdat de TRACE van Simon's BASIC geen single-step, dus geen stapsgewijs programmaverloop mogelijk maakt, is de

programma-controle beperkt. Het scrollen van de programmaregels in het TRACE-venster verloopt zeer snel. Het is daarom zinvol, om voor het testen van het programma op diverse plaatsen wachtlussen in te bouwen. Natuurlijk moeten deze regels in het definitieve programma weer gewist worden. Een wachtlus van ongeveer 1 seconde kan met het Simon's BASIC-commando 'PAUSE 1' worden gecreëerd.

3.1.2 RETRACE

Format	RETRACE
Parameter	geen
Functie	Na het wissen van het beeldscherm wordt het laatst getoonde TRACE-venster teruggehaald.

In het vorige hoofdstuk is al verteld, dat voor het invoeren van opdrachten in de eerste 6 regels van het TRACE-venster het beeldscherm moet worden gewist. Wanneer u nu nog eens wilt kijken naar de laatste TRACE-regels, is het commando RETRACE noodzakelijk. Het volgende programma demonstreert de RETRACE-instructie.

```
 5 PRINT CHR$(147);
10 FOR I=1 TO 10
20 PRINT I
30 NEXT
40 PRINT CHR$(147)
```

De laatste regel van dit programma wist het beeldscherm. Het kan overigens ook in een ander programma voorkomen dat het beeldscherm wordt gewist en de laatste TRACE-regels mee gewist worden. Deze regels kunnen echter juist interessant zijn. Laat dit programma lopen, nadat u de TRACE-modus met 'TRACE 10' hebt ingeschakeld. Na afloop van het programma staan er geen TRACE-regels meer op het scherm. Voer nu het commando 'RETRACE' in, en het volgende TRACE-venster verschijnt op het beeldscherm.

```
#20
#30
```

```
#20
#30
#20
#30
```

Het programma is steeds tussen de regels 20 en 30 heen en weer gesprongen. De eigenlijke lus bevindt zich dus in regels 20 en 30. U ziet, dat de regels 5 en 10 slechts een maal worden doorlopen.

3.1.3 DUMP

Format	DUMP
Parameter	geen
Functie	Toont alle gebruikte variabelen met hun actuele waarde.
Opmerking	Geeft problemen bij negatieve getallen en lege strings.

Na een onderbreking van het programma is het interessant om belangrijke variabelen te controleren. Daarvoor gebruikt men het PRINT-commando. Het nadeel is, dat voor iedere weer te geven variabele een PRINT-commando nodig is. De instructie DUMP vergemakkelijkt deze controle. Hij geeft alle variabelen, die in het programma gebruikt zijn met de desbetreffende waarden weer op het beeldscherm. De output van het commando DUMP kan net als bij de instructie LIST met de CTRL-toets worden vertraagd. Het DUMP-commando lijkt misschien gemakkelijk, maar deze positieve indruk wordt door enige gebreken verdreven.

- 1. er worden geen arrays getoond
- 2. negatieve variabelen worden positief voorgesteld
- 3. bij negatieve gehele getallen wordt de waarde 64 opgeteld
- 4. gedefinieerde lege strings als A$="" worden met 256 niet te definiëren tekens gevuld

Dit zijn de belangrijkste gebreken van het DUMP-commando. Alleen strings en positieve variabelen worden correct weergegeven. Voer bijvoorbeeld eens de volgende regel in in de direkte modus, dus zonder regelnummer.

```
A=10:B=-5:C%=-57:D$=""
```

Voordat u nu het commando DUMP invoert, zou u er eigenlijk bij moeten gaan zitten, want het resultaat is verbazingwekkend:

```
A=10
B=5
C%=65479
D$="????????????????????????????????????
????????????????????????????????????????
????????????????????????????????????????
????????????????????????????????????????
????????????????????????????????????????
????????????????????????????????????????
????????????"
```

Alleen de variabele A wordt correct weergegeven! In dit geval is het verstandig de variabelen met het volgende commando op te vragen.

```
PRINT A,B,C%,D$
```

Het resultaat is dan volkomen correct:

```
10  -5  -57
```

De numerieke variabelen worden gegeven, de lege string niet.

Toch moet u het commando niet uit uw handboek schrappen. Wanneer u bij DUMP rekening houdt met deze "BUGS", dan is de instructie toch - zij het slechts beperkt - te gebruiken. Bij zo'n omvangrijke uitbreiding van uw programmeermogelijkheden als Simon's BASIC geeft, mag u op dergelijke fouten niet te veel kritiek hebben.

3.1.4 COLD

Format	COLD
Parameter	geen
Functie	Brengt Simon's BASIC in de RESET-positie

Wanneer u de Commodore 64 zonder Simon's BASIC wilt terugbrengen naar de toestand na het inschakelen, voert u het commando 'SYS 64738' in. Voert u dit commando in, wanneer Simon's BASIC in de computer aanwezig is, dan krijgt u Simon's BASIC in z'n beginpositie. Het adres 64738 kan gemakkelijk worden vergeten. De opdracht COLD vervult dezelfde functie en is ook gemakkelijker te onthouden. Als u deze instructie invoert, moet u erop letten, dat het programma dat zich in het geheugen bevindt, weer wordt gewist. Ook alle variabelen worden teruggezet. Het programma kan na de koude start weer worden teruggeroepen door het commando 'OLD' (vergelijk 3.1.5).

De koude start met 'COLD' is net als 'SYS 64738' geen koude start in de eigenlijke zin van het woord. Koude start betekent, dat alle adressen in het RAM-geheugen op nul worden gezet. Dat is bij 'COLD' of 'SYS 64738' niet het geval. Hier worden alleen de BASIC-wijzers, die op begin en einde van het programma wijzen, weer in de uitgangspositie teruggebracht. Omdat niet het gehele hoofdgeheugen wordt gewist, kan men hier spreken van een "warme start". Een "echte" koude start is bij de Commodore 64 alleen mogelijk door het uit- en weer inschakelen van de computer.

3.1.5 OLD

Functie	OLD
Parameter	geen
Functie	regenereren van een met NEW gewist programma

Als u per ongeluk een van de commando's NEW of COLD hebt ingevoerd, die het BASIC-programma wissen, is deze instructie erg handig. Dit haalt het gewiste BASIC-programma weer terug. Probeer het eens uit, door eerst de volgende BASIC-regels in te voeren.

```
10 REM ***************
20 REM *    TEST OLD   *
30 REM ***************
```

Wis het programma met het commando 'NEW'. Met een aansluitende 'LIST' kunt u zich ervan vergewissen of het programma ook werkelijk is verdwenen. Gebruik de opdracht 'OLD', om het programma weer terug te halen. Wanneer u 'LIST' invoert, ziet u het resultaat. Hetzelfde voorbeeld kunt u met het commando 'COLD' doorvoeren. Voer deze instructie in en roep daarna het programma met 'OLD' weer terug. Het commando 'OLD' heeft geen effect, wanneer na het commando 'NEW' BASIC-regels worden gewist, veranderd of toegevoegd.

3.1.6 Testopdrachten

1) Welke instructie schakelt de TRACE-modus van Simon's BASIC in?

   A) TRACE
   B) TRACE 0
   C) TRACE 10
   D) RETRACE

2) Welke instructie haalt het laatst getoonde TRACE-venster weer op het gewiste beeldscherm terug?

   A) TRACE
   B) TRACE 0
   C) TRACE 10
   D) RETRACE

3) Wat toont de instructie DUMP op het beeldscherm?

   A) de gebruikte variabelen
   B) de inhoud van de gebruikte variabelen

4) Door welke instructie krijgt u een koude start?

A) COLD
B) OLD
C) NEW

5) Welke instructie haalt een gewist BASIC-programma weer terug?

A) COLD
B) OLD
C) NEW

3.2 Interne foutenbehandeling

De behandeling van de interne fouten in het programma is noodzakelijk om fouten in het programma tegen te gaan en een weigering te verhinderen. Simon's BASIC biedt die mogelijkheid. Of dat nu een 'SYNTAX ERROR' of een 'DEVICE NOT PRESENT ERROR' is, alle fouten worden in het programma herkend en foutmeldingen kunnen apart worden aangegeven. De volgende paragrafen zullen deze omvangrijke mogelijkheden uitvoerig demonstreren.

3.2.1 ON ERROR

Format	ON ERROR:GOTO ln
Parameter	ln - regelnummer waarnaar gesprongen wordt als er een fout optreedt
Functie	een programma voorzien van een foutcontrole
Voorbeeld	ON ERROR:GOTO 1000
Opmerking	De instructie geeft het foutnummer in de variabele ERRN en het regelnummer met de fout in de variabele ERRLN

Een goed georganiseerd en geprogrammeerd programma mag in geen enkel geval weigeren. Daartoe draagt de intensieve controle van de door de gebruiker ingevoerde waarden bij. Wanneer u bijvoorbeeld de 120ste reeks met gegevens van een array wilt hebben, die alleen tot 100 wordt afgemeten, moet het programma dit herkennen en u op de foute invoer attenderen. Is dit niet het geval, dan geeft het programma het bij de adressering van de arrays op, met de melding van de fout '?BAD SUBSCRIPT ERROR IN...'.

Maar niet iedere fout kan door het programma worden verhinderd. Nemen we nu eens aan, dat er gegevens op disk moeten worden opgeslagen, maar u vergeet uw diskdrive aan te sluiten. Dit veroorzaakt dan bij de eerste poging om gegevens over te dragen, de foutmelding '?DEVICE NOT PRESENT ERROR IN ...'. Het is echter onmogelijk, om vanuit het programma vast te stellen of een toestel aangesloten is. Deze fout en nog vele andere kunnen door Simon's BASIC worden herkend. Daarvoor wordt het foutnummer in de variabele ERRN opgeslagen en het regelnummer, waarin de fout wordt herkend, in ERRLN. De foutnummers kunt u ontlenen aan de volgende tabel van de door Simon's BASIC herkenbare fouten.

1	TOO MANY FILES
2	FILE OPEN
3	FILE NOT OPEN
4	FILE NIT FOUND
5	DEVICE NOT PRESENT
10	NEXT WITHOUT FOR
11	SYNTAX
12	RETURN WITHOUT GOSUB
13	OUT OF DATA
14	ILLEGAL QUANTITY
15	OVERFLOW
16	OUT OF MEMORY
17	UNDEFINED STATEMENT
18	BAD SUBSCRIPT
19	RE-DEMENSIONED ARRAY
20	DIVISION BY ZERO
21	ILLEGAL DIRECT
22	TYPE MISMATCH
23	STRING TOO LONG

Alle foutmeldingen van de diskdrive kunnen echter niet door ON ERROR worden herkend. Deze fouten moeten uit het foutenkanaal van de floppy worden gelezen en geanalyseerd. Meer hierover vindt u in "HET GROTE FLOPPY-BOEK" van Data Becker Nederlands*. Maar we gaan over naar het gebruik van de instructie 'ON ERROR'. Deze instructie wordt op de plaats van het programma ingevoerd, waar de foutencontrole wordt overgelaten aan het programma. Mocht u dus uit principe de foutencontrole willen overnemen, dan is dit de eerste regel

van het programma. Een programma behoort echter niet zo onzeker te zijn, dat die vereist is. Slechts een fout, die programma-technisch niet kan worden vermeden, moet op de aangegeven plaats met ON ERROR worden gelokaliseerd. De foutmelding "DEVICE NOT PRESENT" bijvoorbeeld, kunt u niet vermijden. Wordt in het programma een toestel aangesproken dat niet is aangesloten, dan verschijnt deze foutmelding. De programmeur kan hier nu de ON ERROR-functie invoeren en de gebruiker erop wijzen dat het toestel aangesloten of ingeschakeld moet worden. Wanneer u ON ERROR invoert, moet u beslissen, waar de gebruiksroutine moet staan. Stellen we ons nu eens voor, dat u de foutencontrole wilt afgeven aan de routine vanaf regel 10000. De desbetreffende instructie zou dan luiden:

```
10 ON ERROR:GOTO 10000
```

De ON ERROR -instructie mag in geen geval worden ingevoerd in de direkte modus, dat wil zeggen zonder regelnummer, omdat anders ook de foutencontrole van direkt ingevoerde aanwijzingen wordt gegeven. De computer kan na iedere input alleen met RUN/STOP en aansluitend NO ERROR weer naar de uitgangspositie worden teruggebracht. Omdat voor de invoer van ON ERROR ook de instructie NO ERROR nodig is, volgt nu eerst de beschrijving van NO ERROR. In paragraaf 3.2.3 worden dan beide instructies ingeschakeld.

3.2.2 NO ERROR

Format	NO ERROR
Parameter	geen
Functie	geeft de foutencontrole weer over aan het Standaard-BASIC.

Deze instructie onderdrukt niet de foutmeldingen, zoals in het handboek van Simon's BASIC is beschreven, maar geeft de met 'ON ERROR' overgenomen foutencontrole weer terug aan het Standaard-BASIC. Na een foutenbehandeling met 'ON ERROR' is het zonder meer noodzakelijk om voor het verlaten van het programma of

na het onderbreken van het programma met de toets RUN/STOP, terug te schakelen door middel van NO ERROR. Wordt dit genegeerd, dan kan dat verrassende gevolgen hebben en zelfs Simon's BASIC ten val brengen. Het zou zinvoller zijn, de ON ERROR-modus buiten een programma helemaal buiten werking te stellen. Het is volkomen zinloos als na de invoer van een instructie, bijvoorbeeld LIST, een vertakking naar de foutroutine van een programma plaatsvindt en alleen de toets RUN/STOP het READY van het bedrijfssysteem afdwingt. Hoe en waar de instructie NO ERROR wordt ingevoerd, leest u in het volgende hoofdstuk.

3.2.3 ON ERROR/NO ERROR in detail

Hoe voert u de instructies ON ERROR en NO ERROR en de variabelen ERRN en ERRLN in het programma in? Op welke belangrijke voorwaarden moet u letten, om vervelende reacties op foutief gebruik te vermijden? De ON ERROR-functie moet weliswaar altijd alleen op de aangegeven plaats worden ingevoerd, maar voor de demonstratie nemen we in de volgende voorbeelden in beginsel de hele foutenanalyse over. Daarvoor wordt de ON ERROR-instructie aan het begin van het programma gebruikt. De desbetreffende regel ziet er dan als volgt uit:

```
10 ON ERROR:GOTO 10000
```

Deze regel bepaalt nu, dat bij het optreden van een fout een vertakking naar regel 10000 plaatsvindt. Bij deze beginregel sluit het eigenlijke programma aan. Voer hier ter illustratie de volgende gedeeltelijk foute regels in.

```
20 REM ==================
30 REM    TEST ON ERROR
40 REM ==================
50 PRINT "---TEST BEGIN---"
60 PRINT "START"
70 PRINT CHR$(256)
80 A=10:A$="20"
90 PRINT A+A$
100 X$(11)="---TEST EINDE---"
110 PRINT X$(11)
120 NO ERROR:END
```

Dit is het programma, dat van enige fouten is voorzien. De foutenanalyse die, zoals bepaald in regel 10, vanaf regel 10000 begint, ontbreekt nog. Een eenvoudige foutenanalyse ziet er ongeveer als volgt uit:

```
10000 REM ********************
10010 REM *    FOUT ANALYSE    *
10020 REM ********************
10030 PRINT"FOUT NUMMER" ERRN "IN REGEL" ERRLN
10040 PRINT"STOPPEN V/H PROGRAMMA"
10050 NO ERROR:END
```

Dit is in eerste instantie een zeer eenvoudige mogelijkheid om op fouten te reageren. Start het programma met 'RUN'. Dan wordt op het beeldscherm weergegeven:

```
---TEST BEGIN---
FOUT NUMMER 11 IN REGEL 60
PROGRAMMAONDERBREKING
```

Zoals u uit de tabel in paragraaf 3.2.1 kunt lezen, wijst het foutnummer 11 op de fout 'SYNTAX ERROR'. Dit nummer wordt in de variabele ERRN opgeslagen. De regel die de fout heeft veroorzaakt, kan uit de variabele ERRLN worden gehaald. Na weergave van het foute nummer en de verkeerde regel wordt het programma hier afgebroken. Dit is echter niet de bedoeling. Op deze manier wordt de fout ook door het Standaard-BASIC behandeld. Het programma moet alleen de melding van de fout aangeven en vervolgens verdergaan. Dat wil zeggen, dat de foutroutine na het aantonen van de fout weer moet verspringen naar de regel, die op de foute regel volgt. Deze regel kan worden berekend. Wanneer het programma continu met de stapgrootte 10 is genummerd, komt het nummer van de volgende regel voort uit het nummer van de foute regel (ERRLN) plus de stapgrootte 10. Dit vereist een berekende sprong. De GOTO-instructie van het standaard-BASIC kan alleen naar een bepaalde regel springen. Simon's BASIC biedt een instructie voor een berekende sprong. U kunt hier een variabele of een rekenkundige uitdrukking als sprongadres invoeren. In ons geval zou deze sprong bij een stapgrootte van 10 als volgt worden gecodeerd:

```
CGOTO (ERRLN+10)
```

De instructie voor de berekende sprong luidt 'CGOTO' en wordt in paragraaf 5.2.5 nader uitgelegd. Het sprongadres komt dus voort uit (ERRLN+10). Voegen we deze instructie in, dan ontstaat de volgende foutroutine:

```
10000 REM ********************
10010 REM *    FOUT ANALYSE    *
10020 REM ********************
10030 PRINT"FOUT NUMMER" ERRN "IN REGEL" ERRLN
10040 PRINT"SPRONG NAAR REGEL" ERRLN+10
10050 CGOTO (ERRLN+10)
```

De regels 10040 en 10050 zijn hier veranderd. Regel 10040 geeft de regel aan, waarin het programma wordt voortgezet. In regel 10050 wordt dan naar deze regel overgesprongen. Nadat u dit programma hebt gestart, wordt het volgende overzicht van de fouten op het beeldscherm weergegeven:

```
FOUT NUMMER 11 IN REGEL 60
SPRONG NAAR REGEL 70
FOUT NUMMER 14 IN REGEL 70
SPRONG NAAR REGEL 80
FOUT NUMMER 22 IN REGEL 90
SPRONG NAAR REGEL 100
FOUT NUMMER 18 IN REGEL 100
SPRONG NAAR REGEL 110
FOUT IN REGEL 110
SPRONG NAAR REGEL 120
```

Dit is een lijst van fouten, waarin de numeriek aangegeven fouten nog met behulp van de tabel in paragraaf 3.2.1 moeten worden vertaald. De fouten van het bedrijfssysteem zijn echter aan de gebruiker van het programma niet bekend. Alle fouten, die u kunt oproepen, zouden door het programma met bijvoorbeeld ON ERROR moeten worden opgevangen. Wanneer een fout wordt ontdekt, moet een foutmelding worden gegeven, die u in duidelijke tekst op uw fout wijst en die u zonodig verdere aanwijzingen geeft. Dit voorbeeld geeft alleen de foutmeldingen aan. Om u te tonen, dat ieder verkeerd nummer door een willekeurige tekst kan worden vervangen, volgt een foutroutine, die de fouten in het Nederlands vertaalt. In plaats van de Nederlandse foutmelding kunt u ook gedetailleerde beschrijvingen van fouten geven, die ook voor u veelzeggend zijn.

Voor wie niet uitgesproken lui is in het schrijven, volgt de betreffende verandering van het programma. We roepen aan het begin van het programma een subprogramma op, dat de foutmeldingen in een array neerlegt. Het subprogramma moet de beginregel 20000 bevatten en in regel 5 worden opgeroepen. Kijk eerst eens naar het complete, veranderde programma.

```
    5 GOSUB 20000
   10 ON ERROR: GOTO 10000
   20 REM ******************
   30 REM ** TEST ON ERROR **
   40 REM ******************
   50 PRINT"---BEGIN V/D TEST---"
   60 PRINT:"START"
   70 PRINT CHR$(256)
   80 A=10:A$="20"
   90 PRINT A+A$
  100 X$(11)="---EINDE V/D TEST---"
  110 PRINT X$(11)
  120 NO ERROR:END
10000 REM *****************
10010 REM ** FOUT ANALYSE **
10020 REM *****************
10030 PRINT FE$(ERRN)"IN REGEL" ERRLN
10040 PRINT"SPRONG NAAR REGEL" ERRLN+10
10050 CGOTO (ERRLN+10)
20000 REM ********************
20010 REM ** FOUTTABEL LEZEN **
20020 REM ********************
20030 DIM FE$(23)
20040 READ I,I$
20050 IF I=0 THEN RETURN
20060 FE$(I)=I$
20070 GOTO 20040
20080 DATA  1,TE VEEL FILES
20090 DATA  2,FILE OPEN
20100 DATA  3,FILE NIET OPEN
20110 DATA  4,FILE NIET GEVONDEN
20120 DATA  5,DEVICE NIET AANWEZIG
20130 DATA 10,'NEXT' ZONDER 'FOR'
20140 DATA 11,SYNTAX FOUT
20150 DATA 12,'RETURN' ZONDER 'GOSUB'
20160 DATA 13,TE WEINIG DATA
20170 DATA 14,ONTOELAATBARE WAARDE
20180 DATA 15,OVERLOOP
20190 DATA 16,TE WEINIG BESCHIKBARE GEHEUGENRUIMTE
20200 DATA 17,MISSENDE REGEL
20210 DATA 18,ONGELDIGE INDEX
20220 DATA 19,VELD AL GEDIMENSIONEERD
20230 DATA 20,DELING DOOR NUL
20240 DATA 21,DIREKT-MODUS VERBODEN
20250 DATA 22,VERKEERD TYPE VARIABELE
20260 DATA 23,KARAKTERREEKS TE LANG
20270 DATA  0, END
```

Een programma met een dergelijke foutenbehandeling kan niet meer weigeren, wanneer een fout voorkomt. Het betekent dus een bepaalde programmabescherming, omdat een afgebroken programma door de gebruiker kan worden gelist of opgeslagen. Voor lezers, die dit programma niet volledig overzien, volgt een beschrijving van de veranderingen. Eerst wordt in regel 5 het subprogramma opgeroepen, dat de Nederlandse foutmeldingen inleest. Dit subprogramma definieert de regels 20000 tot 20270. In regel 20030 wordt de array ingericht, die de foutmeldingen bevat. Hierbij moet de index tegelijkertijd het foutnummer zijn. Dat wil zeggen, de tekst voor foutnummer 11 bevindt zich in string FE$(11) enzovoort. Regel 20040 leest uit de DATA-regels het foutnummer en de duidelijke tekst. De tekst van de fourmelding wordt dan in de string met de index van het foutnummer opgeslagen. Dit herhaalt zich, totdat het einde van de DATA-regels is bereikt, wat door het lezen van een nul wordt herkend. Dan wordt teruggesprongen naar het hoofdprogramma (RETURN). De foutmeldingen bevinden zich dus nu in de array. In de foutenanalyse wordt de foutmelding aangegeven. Het foutnummer (ERRN) is index van de desbetreffende string. Regel 10020 moet navenant worden veranderd.

Start door middel van RUN nog eens dit programma, dat met een erg comfortabele foutenanalyse is uitgerust. Er wordt een lijst met fouten gegeven, die aanzienlijk meer zegt dan eerder het geval was:

```
--- TEST BEGIN ---


SYNTAXIS FOUT IN REGEL 60
SPRONG NAAR REGEL 70
ONTOELAATBARE WAARDE IN REGEL 70
SPRONG NAAR REGEL 80
VERKEERD TYPE VARIABELE IN REGEL 90
SPRONG NAAR REGEL 100
ONGELDIGE INDEX IN REGEL 100
SPRONG NAAR REGEL 110
ONGELDIGE INDEX IN REGEL 110
SPRONG NAAR REGEL 120
```

In de ontwikkelingsfase van uw programma is zo'n foutenanalyse absoluut bruikbaar. De foutmeldingen worden echter ook op het beeldscherm gegeven, wanneer het misschien niet gewenst is, bijvoorbeeld in een schermmasker. Het is zinvoller en professioneler deze lijst met fouten tijdens het programmaverloop op de printer of naar een bestand over te brengen. Het beeldscherm wordt dan niet beïnvloed. De veranderingen zijn gemakkelijk in te bouwen.

foutprotocol op de printer:

- printerkanaal in het begin van het programma openen :
  3 OPEN 1,4
- regels 10030 en 10040 veranderen:
  10030 PRINT#1,FE$(ERRN)" IN REGEL"ERRLN
  10040 PRINT#1,"SPRING NAAR REGEL"ERRLN+10
- printerkanaal aan het eind van het programma sluiten:
  120 NO ERROR:CLOSE 1:END

foutprotocol in bestand:

- regel 3 toevoegen:
  3 OPEN 1,8,2,"FOUTENBESTAND,S,W"
- regels 10030 en 10040 als boven veranderen
- bestand als boven sluiten

Na afloop van het programma hebt u de lijst met fouten of zwart-op-wit op papier, of in de vorm van een bestand op diskette. Dit bestand kunt u naar behoefte op het beeldscherm brengen. De volgende BASIC-regels nemen deze taak over:

```
10 OPEN 1,8,2,"FOUTENBESTAND,S,R"
20 GET#1,X$
30 PRINT X$;
40 IF ST<>64 THEN 20
50 CLOSE 1
```

Hier wordt het sequentiële foutenbestand geopend en bytes-gewijs uitgelezen; het wordt eveneens op het beeldscherm weergegeven. Het einde van het bestand wordt aan de statusvariabele ST herkend, die dan de waarde 64 heeft. Nu staan alle professionele wegen tot foutenanalyse van uw programma voor u open. Profiteer zo veel mogelijk van deze faciliteiten, die Simon's BASIC u biedt.

3.2.4 OUT

Op deze plaats zou de instructie OUT moeten worden beschreven. Er wordt echter vanaf gezien in uw belang, omdat die instructie in de

aanwezige diskette-versie van Simon's BASIC niet kan functioneren. De beschrijving van de instructie die in het handboek staat, geldt voor de instructie NO ERROR. Wordt die instructie in het programma gebruikt, dan heeft dit hetzelfde effect als de instructie END. In de direct-mode geeft het de laatst gegeven melding van de computer aan. Voert u bijvoorbeeld RUN in, wat een '?SYNTAX ERROR' veroorzaakt, dan wordt deze foutmelding met een aansluitend OUT nog een keer uitgevoerd. De instructie kan dus op geen enkele manier worden gebruikt.

3.2.5 Testopdrachten

1) Er moet een foutcontrole in het programma worden aangebracht. De routine daarvoor begint op regel 1000. Welke instructie vervult de opgave?

   A) ON ERROR: GOTO 1000
   B) ON ERROR; GOTO 1000
   C) ON ERROR, GOTO 1000
   D) ON ERROR GOTO 1000

2) In welke variabele wordt door de instructie ON ERROR het nummer van de foutcodering beschikbaar gesteld?

   A) ER
   B) EN
   C) ERN
   D) ERRN

3) In welke variabele wordt de foutmeldingsregel beschikbaar gesteld?

   A) ERRLINE
   B) ERLN
   C) ERRLN
   D) ERL

4) Welke instructie brengt de foutcontrole weer onder standaard-BASIC?

   A) OUT ERROR

B) NO ERROR
C) OFF ERROR

5) Welke van de volgende foutmeldingen kan niet met de instructie ON ERROR worden opgevangen?

A) OUT OF MEMORY ERROR
B) OUT OF DATA ERROR
C) LOAD ERROR

4 BESCHERMING VAN HET PROGRAMMA

4.1 DISAPA

Format	DISAPA
Parameter	geen
Functie	Met SECURE 0 te beveiligen regels merken

Dit commando moet u als eerste opdracht voor iedere regel zetten, die vervolgens door SECURE 0 moet worden beschermd. Laten we bijvoorbeeld eens aannemen, dat regel 20 van het volgende programma moet worden beschermd.

```
10 INPUT"VOER DE PROGRAMMA CODE IN";A$
20 IF A$="A12B13" THEN 40
30 PRINT"CODE IS NIET JUIST!":GOTO 10
40 PRINT"CODE IS JUIST!"
```

Daarvoor moet u het commando 'DISAPA' voor de eigenlijke code van regel 20 zetten.

```
20 DISAPA:IF A$="A12B13"THEN 40
```

Wanneer u het programma nu door LIST op het beeldscherm brengt, zult u constateren, dat achter het DISAPA-commando vier dubbele punten zijn geplaatst.

```
10 INPUT"VOER DE PROGRAMMA CODE IN";A$
20 DISAPA::::IF A$="A12B13" THEN 40
30 PRINT"CODE IS NIET JUIST!":GOTO 10
40 PRINT"CODE IS JUIST!"
```

Wanneer het volledige programma moet worden beschermd, moet iedere regel van een DISAPA-commando worden voorzien. Dat is natuurlijk niet zo gemakkelijk, maar het is jammer genoeg niet te vermijden. Het doel van DISAPA is echter, belangrijke regels te beschermen, die bijvoorbeeld een kopieerbescherming bevatten. Regels, die met DISAPA zijn gemerkt, worden door middel van het volgende commando beschermd.

4.2 SECURE

Format SECURE 0

Parameter geen

Functie Beveiligen van met DISAPA gemerkte regels

Het commando SECURE 0 wordt altijd in de direkte modus, dus buiten een programma om ingevoerd. Regels, die van tevoren door DISAPA werden gemerkt, worden hier beschermd. In paragraaf 4.1 werd regel 20 van het volgende programma met DISAPA gemerkt.

```
10 INPUT"VOER DE PROGRAMMA CODE IN";A$
20 DISAPA:::IF A$="12B13" THEN 40
30 PRINT"CODE IS NIET JUIST!":GOTO 10
40 PRINT"CODE IS JUIST!"
```

Voer nu het commando SECURE in en u zult merken, dat regel 20 wordt beschermd.

```
20
```

De inhoud van regel 20 is dus onherkenbaar gemaakt. Dit heeft het nadeel, dat regel 20 natuurlijk niet meer kan worden teruggehaald. Dat is ook logisch; het programma zou immers niet beschermd zijn, wanneer het naar behoefte kon worden opgevraagd. De beschermde regels kunnen alleen nog worden gewist. Het is dus aan te raden, van ieder beschermd programma twee versies in het geheugen op te slaan.

- 1. de versie met de DISAPA-instructies;
- 2. de versie volgens SECURE 0.

Dan kunt u altijd het onbeschermde programma laden, veranderen en weer beveiligen.

Misschien zult u zich afvragen, hoe een regel onzichtbaar wordt en desondanks toch werkzaam is. De truc ervan is, dat aan een BASIC-regel een byte met de inhoud 0 en 4 dubbele punten voorafgaan. Deze regel wordt op grond van de 0 niet gelist, maar door de 4 dubbele punten toch verwerkt. We zouden hier een programma kunnen laten zien, dat de regels, die met SECURE 0 zijn beschermd, weer zichtbaar maakt, maar dan zou dit commando zinloos zijn. Dat karwei laten we aan de programmeur over, die graag moeilijke vraagstukken oplost.

5 STRUCTUUR VAN HET PROGRAMMA

De structuur van een programma bepaalt tegelijkertijd de overzichtelijkheid. Het is al in Standaard-BASIC zinvol om gestructureerd te programmeren. Doelmatig is bijvoorbeeld, om subprogramma's aan het einde van het programma in te richten en met REMARKS (REM) te kenmerken. Verder zou een vervlechting van het programma kenbaar gemaakt moeten worden. Daarvoor verschuift u de regels, die bij de desbetreffende vervlechting horen, enige plaatsen naar rechts. Dit kunt u uitsluitend bereiken met dubbele punten of met een dubbele punt en een spatie. Bij een lus wordt de vervlechting bijvoorbeeld als volgt gekenmerkt:

```
10 FOR I=1 TO 10
20 ::::A=A+1
30 ::::PRINT I,A
40 NEXT I
```

Een gestructureerde IF-opvraag is eveneens overzichtelijker:

```
10 IF A$="PRINTEN" THEN 60
20 :    PRINT"GEEN PRINTERUITVOER"
30 :    PRINT"UITVOER OP BEELDSCHERM (J/N)
40 :    GET X$:IF X$<>"N" AND X$<>"J" THEN 40
50 :    IF X$="J" THEN A$="BEELDSCHERMUITVOER"
60 GOSUB 1000
```

Het is nu gemakkelijker om te herkennen, wat er in de NEE-tak van IF-test gebeurt. De mogelijkheden van het structureren zijn echter gering met Standaard-BASIC. Simon's BASIC biedt vele mogelijkheden, om het programma duidelijk vorm te geven.

5.1 Logische operaties

5.1.1 IF...THEN...ELSE

Format IF voorwaarde THEN ja-tak : ELSE : nee-tak

Parameter voorwaarde - logische vergelijking

ja-tak (aanwijzing indien positief)

nee-tak (aanwijzing indien negatief)

Voorbeeld IF X$="N" THEN PRINT "EINDE":END:ELSE:GOTO 1000

Deze IF-test maakt het mogelijk zowel de ja-vertakking als de nee-vertakking van een voorwaarde in een regel af te handelen. Met de traditionele IF-test worden alleen de aanwijzingen voor de ja-vertakking in de IF-regel opgenomen. Vergelijk de beide mogelijkheden met het volgende voorbeeld: een teken moet via het toetsenbord worden ingelezen. Bij het bedienen van toets 'D' moet het subprogramma vanaf regel 1000 en bij 'B' het subprogramma vanaf regel 2000 worden opgeroepen. Met Standaard-BASIC lossen we het probleem met de volgende programmaregels op.

```
10 GET X$:IF X$<>"P" AND X$<>"B" THEN 10
20 IF X$="P" THEN GOSUB 1000
30 IF X$="B" THEN GOSUB 2000
40 ....
```

Hier waren voor de vertakking twee IF-regels nodig. Het probleem kan echter ook door een enkele IF-test worden opgelost.

```
10 GET X$:IF X$<>"P" AND X$<>"B" THEN 10
20 IF X$="P" THEN GOSUB 1000:GOTO 40
30 GOSUB 2000
40 ......
```

Ook hier zijn 4 regels nodig. Simon's BASIC geeft de volgende mogelijkheid voor een oplossing.

```
10 GET X$:IF X$<>"P" AND X$<>"B" THEN 10
20 IF X$="P" THEN GOSUB 1000:ELSE:GOSUB 2000
30 ....
```

Deze oplossing is mooier en moet voor zover mogelijk, altijd worden toegepast.

5.1.2 RCOMP

Format	RCOMP: ja-tak : ELSE : nee-tak
Parameter	ja-tak (aanwijzing indien positief)
	nee-tak (aanwijzing indien negatief)
Functie	test op de laatst voorgekomen voorwaarde bij IF ... THEN ... ELSE ...
Voorbeeld	RCOMP END:ELSE:GOTO 1000

Het kan in een programma zeker voorkomen, dat na een IF...THEN...ELSE-test en enige andere commando's dezelfde voorwaarde nog eens moet worden opgevraagd. Hiervoor is dan het iets kortere commando RCOMP, dat als verkorte vorm van het laatste IF...THEN...ELSE commando dezelfde test als inhoud heeft. Een voorbeeld: het nummer van ieder artikel van het bestand "ARTDAT" wordt overgenomen in de variabele AN. Is het artikelnummer groter dan 999, dan moet het leveranciersnummer (LN) van het artikel in het bestand "LEV.1000-" worden opgenomen, anders in het bestand "LEV.-999". Verder wordt onder dezelfde voorwaarden de prijs (PR) in de bestanden "PRIJS.1000-" en "PRIJS.1000" overgenomen. Het volgende programma lost dit probleem op.

```
100 REM ********************
105 REM *  HOOFDPROGRAMMA   *
110 REM ********************
120 OPEN 1,8,2,"ARTDAT,S,W"
130 INPUT#1,AN,LN,PR
140 IF AN>999 THEN GOSUB 1000:ELSE:GOSUB 2000
150 RCOMP GOSUB 3000:ELSE:GOSUB 4000
170 IF ST<>64 THEN 130
180 CLOSE 1:END
1000 REM ***************
1005 REM BOEKEN  LEV.1000
1010 REM ***************
...
...
1040 RETURN
2000 REM ***************
2005 REM BOEKEN  LEV.-999
2010 REM ***************
2020 ...
```

```
2030 ...
2040 RETURN
3000 REM *****************
3005 REM BOEKEN  PRIJS.1000
3010 REM *****************
3020 ...
3030 ...
3040 RETURN
4000 REM *****************
4005 REM BOEKEN  PRIJS.-999
4010 REM *****************
4020 ...
4030 ...
4040 RETURN
```

Zonder gebruik van het commando 'RCOMP' zou de test in regel 140 moeten worden herhaald. De aanwijzing 'RCOMP' kan zolang herhaald worden, tot er een andere IF...THEN...ELSE-verbinding volgt. Omdat het commando RCOMP alleen de voorwaarde van het laatste IF...THEN...ELSE-commando vervangt, kunnen de aanwijzingen voor de ja- en nee-vertakking natuurlijk verschillend zijn. De overzichtelijkheid van het programma kan verloren gaan wanneer de laatste IF...THEN...ELSE-test van een RCOMP-commando in een ander programma-onderdeel of ver van te voren geplaatst is. Het gebruik van RCOMP is dus alleen zinvol vlak bij de desbetreffende IF...THEN...ELSE-test.

5.1.3 REPEAT...UNTIL

Format	REPEAT : lus : UNTIL : voorwaarde
Parameter	lus - instructies in een lus
	voorwaarde - voorwaarde waaraan voldaan moet worden om de lus af te sluiten
Functie	lusbesturing met een voorwaarde
Voorbeeld	REPEAT: A=A+1: UNTIL A=10000

Dit is een alternatieve inrichting van de lus in vergelijking met de traditionele FOR...NEXT-lussen. Door het gebruik van dit alternatief kan een overzichtelijke lus worden opgebouwd. Het volgende voorbeeld zal dit verduidelijken. De som van de getallen 10-100 moet worden gevormd.

```
10 A=10
20 REPEAT
30 ::::S=S+A
40 ::::A=A+1
50 UNTIL A > 100
60 PRINT S
```

Na afloop van het programma wordt het resultaat (5005) weergegeven. Het is u wellicht opgevallen, dat er werd opgevraagd door middel van 'A > 100' en niet 'A=100'. Deze voorwaarde is gewenst, omdat de teller (A) na de optelling wordt verhoogd. De laatst bereikte 100 moet er dus nog bij opgeteld worden. Dit voorbeeld kan echter veel beter en sneller worden opgelost met de gewone FOR...NEXT-methode.

```
10 FOR A=10 TO 1000
20 ::S=S+A
30 NEXT
40 PRINT S
```

Eigenlijk is dit niet zo'n goed voorbeeld. Het commando FOR...NEXT brengt het er veel beter van af. Het belangrijkste voordeel van REPEAT...UNTIL ligt in het sturen van de lussen met een onzeker einde. Dat wil zeggen, dat het FOR...NEXT-commando in de tweede parameter (TO...) altijd een bekend getal bevat. Het einde van een FOR...NEXT lus is dus van te voren bepaald en niet afhankelijk van gebeurtenissen binnen de lus. Bij de REPEAT-UNTIL-sturing is het eindcriterium echter een willekeurige voorwaarde, waarvan verwacht wordt dat die eens binnen de lus wordt vervuld. Daarom is REPEAT...UNTIL niet vergelijkbaar met FOR...NEXT. REPEAT...UNTIL is eigenlijk een alternatief voor aftasten binnen opgebouwde lussen. Dit wordt duidelijk aan de hand van een eenvoudig voorbeeld: er moet een sequentieel bestand worden gelezen. Het einde van het bestand wordt herkend aan het feit, dat de statusvariabele ST door de computer met de waarde 64 wordt gedefinieerd. Dit einde van het bestand is onzeker, omdat we niet weten hoe lang het bestand zal zijn. Daardoor kan het einde niet met een FOR...NEXT-lus worden bepaald. Met gewoon BASIC zou men het bestand als volgt uitlezen:

```
10 OPEN 1,8,2,"BESTAND,S,R"
20 I=0
30 ::::I=I+1
40 ::::INPUT#1,A$(I)
50 IF ST<> 64 THEN GOTO 30
60 CLOSE 1:END
```

Hier wordt een lus opgebouwd, die wordt verlaten met een IF-test. Met REPEAT...UNTIL kan dit eleganter worden opgelost.

```
10 OPEN 1,8,2,"BESTAND,S,R"
20 I=0
30 REPEAT
40 ::::I=I+1
50 ::::INPUT#1,A$(I)
60 UNTIL ST<>64
70 CLOSE 1
```

Hier is weliswaar een extra regel nodig, maar er ontstaat een belangrijke bijdrage tot een gestructureerde programmering: een GOTO kan worden vermeden.

5.1.4 LOOP...EXIT IF...END LOOP

Format	LOOP : lus : EXIT IF voorwaarde : END LOOP
Parameter	lus - instructies in een lus
	voorwaarde - voorwaarde voor beëindigen van de lus
Functie	lusbesturing met voorwaarde
Voorbeeld	LOOP:A=A+1:EXIT IF A=1000:END LOOP

Op het eerste gezicht lijkt deze werkwijze met lussen erg op REPEAT...UNTIL. De essentiële voordelen van deze besturing zijn echter:

- De voorwaarde voor het afbreken van de lus kan midden in de lus-aanwijzingen aanwezig zijn. Voorbeeld:

```
100 LOOP
110 A=A+1
120 EXIT IF A>20
130 B=B+A
140 END LOOP
```

- Er kunnen verschillende voorwaarden met betrekking tot het afbreken van een lus op een willekeurige plaats worden ingevoerd. Voorbeeld:

```
100 LOOP
110 A=A+1
120 EXIT IF A>20
130 B=B+A
140 EXIT IF B>100
150 END LOOP
```

Laten we nog eens naar de gestructureerde programmering kijken. Om het laatste voorbeeld met Standaard-BASIC op te lossen, zijn de volgende regels noodzakelijk.

```
110 A=A+1
120 IF A>20 THEN 150
130 B=B+A
140 IF A>100 THEN 150
150 ....
```

Hier zijn weer twee GOTO-commando's nodig, wat niet de bedoeling is van de gestructureerde programmering.

Een ander voorbeeld voor het gebruik van LOOP: via het toetsenbord moet worden gevraagd of een print gewenst is. Zo ja, dan moet het subprogramma 1000 worden opgeroepen; in het andere geval moet het programma worden voortgezet. Eerst de oplossing met Standaard-BASIC:

```
100 PRINT"PRINTER UITVOER (J/N)?"
110 GET X$:IF X$<>"N" AND X$<>"J" THEN 110
120 IF X$="J" THEN GOSUB 1000
```

```
130 ....
```

Nu de oplossing met Simon's BASIC:

```
100 PRINT"PRINTER UITVOER (J/N)?"
110 LOOP
120 GET X$
130 EXIT IF X$="J"
140 EXIT IF X$="N"
150 END LOOP
160 IF X$="J" THEN GOSUB 1000
170 ....
```

Hier zijn voor de oplossing van het probleem weliswaar meer regels nodig, maar dit programma is dan ook overzichtelijker. Ook valt weer een GOTO weg, wat de duidelijkheid van het programma ten goede komt. "Wilde sprongen" van de ene naar de andere regel zijn niet de bedoeling van een gestructureerd programma.

5.1.5 Testopdrachten

1) Welke van de volgende IF...THEN...ELSE-toepassingen is correct?

   A) IF A=1 THEN 100:ELSE GOTO 200
   B) IF A=1 THEN 100:ELSE:GOTO 200
   C) IF A=1 THEN 100 ELSE GOTO 200
   D) IF A=1 THEN 100 ELSE:GOTO 200

2) Waarop heeft de instructie RCOMP betrekking?

   A) op de laatste IF-test
   B) op de laatste IF...THEN...ELSE-test
   C) op de eerste IF-test
   D) op de eerste IF...THEN...ELSE-test

3) Welke REPEAT-toepassing is de juiste?

   A) REPEAT:GETX$:UNTIL X$<>""

B) REPEAT GETX$ UNTIL X$<>""
C) REPEAT GETX$:UNTIL X$<>""

4) Welke LOOP-toepassing is verkeerd?

A) LOOP:GET X$:EXIT IF X$="J":EXIT IF X$="N" END LOOP
B) LOOP:A=A+11:B=B+A:EXIT IF A>10 OR B>20:END LOOP
C) LOOP:EXIT IF I=10:I=I+1:READ A(I):END LOOP

5) Welke getallen worden door het volgende programma aangegeven?
A=10:LOOP:EXIT IF A>20:A=A+1:PRINT A:END LOOP

A) 10 tot 20
B) 11 tot 20
C) 10 tot 21
D) 11 tot 21

5.2 Behandeling van de sprong

5.2.1 PROC

Format	PROC label
Parameter	label - symbolisch adres
Functie	symbolisch sprongadres toekennen
Voorbeeld	100 PROC INVOERROUTINE

Bij de beschrijving van het commando RENUMBER werd verwezen naar dit hoofdstuk. RENUMBER kan geen rekening houden met GOTO- of GOSUB-regelnummers die parallel lopen aan de programma-nummering. Dit kunt u alleen vermijden, wanneer deze beide commando's niet worden gebruikt. Een programma zonder GOTO en GOSUB is onmogelijk, zult u denken. Maar dat is niet zo. Voor beide

commando's zijn er alternatieven, die geen bepaald regelnummer maar symbolische adressen gebruiken. Regels, die met deze commando's moeten worden opgeroepen, moeten van te voren van een symbolisch adres (een label) worden voorzien. Hiervoor is er het commando PROC, dat zonder problemen kan worden toegepast. Voor iedere regel, die symbolisch wordt opgeroepen, wordt een PROC-regel gezet. Een voorbeeld:

```
100 PROC [redacted] INVOEREN
110 INPUT"VOERT U EEN NAAM IN:";X$
```

Wanneer nu regel 110 moet worden opgeroepen, gebeurt dit niet zoals gewoonlijk door middel van 'GOTO 110', maar door het commando 'CALL ~~INPUT NAME~~', dat later wordt beschreven.
INVOEREN

Opgelet: In de PROC-regel mogen zich geen andere aanwijzingen bevinden!

5.2.2 END PROC

Format	END PROC
Parameter	geen
Functie	een routine afsluiten
Opmerking	vergelijkbaar met RETURN

Zoals gezegd worden met PROC symbolische sprongadressen toegekend. Bovendien kan samen met het commando END PROC een volledige routine worden gedefinieerd. Het begin van de routine bevat dan de PROC-regel en het einde van de routine de END PROC-regel. Bijvoorbeeld:

```
100 PROC UITVOER ADRESSEN
110 ::::PRINT"NAAM:      "N$
120 ::::PRINT"VOORNAAM   "V$
130 ::::PRINT"STRAAT     "S$
```

```
140 ::::PRINT"PLAATS     "P$
150 END PROC
```

Dit is een overzichtelijk opgebouwde routine, die met het commando EXEC UITVOER ADRES kan worden opgeroepen. Het commando END PROC komt overeen met het bekende commando RETURN.

5.2.3 CALL

Format	CALL label
Parameter	label - symbolisch sprongadres
Functie	sprong naar een met PROC gedefinieerd label
Voorbeeld	CALL INVOER
Opmerking	vergelijkbaar met GOTO

Dit commando roept de label op, die met PROC is gedefinieerd. Uit een GOTO 100 ontstaat nu bijvoorbeeld een CALL ADR1. Een programma, dat van een label is voorzien, kan er nu bijvoorbeeld als volgt uitzien:

```
100 PROC GET
110 ::::GET X$:IF X$="" THEN CALL GET
120 ::::IF X$="A" THEN CALL INPUTA:ELSE:CALL GET
130 PROC INPUTA
    ...
```

En weer werden alle GOTO's vermeden. Het nadeel van RENUMBER van Simon's BASIC wordt steeds meer naar de achtergrond verdrongen.

5.2.4 EXEC

Format	EXEC label
Parameter	label - symbolisch sprongadres
Functie	oproep van een met PROC gedefinieerde routine
Voorbeeld	EXEC INVOERROUTINE
Opmerking	vergelijkbaar met GOSUB

Routines (subprogramma's) zijn in ieder groter programma noodzakelijk. Gestructureerde programmeertalen zoals COBOL of PASCAL bestaan bijna uitsluitend uit routines, die door het hoofdprogramma worden gestuurd. Deze mogelijkheid bestaat nu ook voor Simon's BASIC. Een goed georganiseerd en gestructureerd programma ziet er ongeveer zo uit:

```
110 EXEC OPEN-BESTAND
120 PROC START
130 ::::EXEC AANWIJZEN INVOERMASKER
140 ::::EXEC VULLEN INVOERMASKER
150 ::::EXEC SCHRIJVEN GEGEVENS
160 ::::PRINT "NOG MEER GEGEVENS (J/N)?"
170 ::::EXEC PROC GET J/N
180 ::::IF AN$="J" THEN CALL START
190 CALL CLOSE-BESTAND
200 EXEC PROGRAMMA-EINDE
```

In dit programma zijn gegevens geregistreerd, waarbij alleen nog de desbetreffende routines moeten worden ontwikkeld. Een programmeerstijl, die voor u zeker ongebruikelijk is, maar toch best de moeite waard. U ontwikkelt eerst het hoofdprogramma en dan de individuele subroutines. Deze gestructureerde programmeerstijl vergemakkelijkt ook het opsporen van fouten. Bij de symbolische adressering zijn er twee foutmeldingen, die u in de gaten moet houden:

PROC NOT FOUND

er is met CALL of EXEC een label aangegeven, dat niet is gedefinieerd.

END PROC WITHOUT EXEC

het programma stuit op END PROC, hoewel geen

EXEC werd uitgevoerd.

5.2.5 CGOTO

Format	CGOTO
Parameter	berekening - rekenkundige expressie
Functie	sprong naar een te berekenen regelnummer
Voorbeeld	CGOTO (N+I*2)
	Springt naar de regel die het nummer heeft dat overeenkomt met het getal dat de berekening oplevert.

Bij de beschrijving van het commando ON ERROR werd u reeds geconfronteerd met deze opdracht. Daar werd de terugsprong van de foutroutine berekend uit de foutregel plus stapgrootte. Zonder het commando CGOTO zou deze berekende terugsprong niet mogelijk zijn. Waar kan men dit commando voor gebruiken? Een mogelijkheid is, na de weergave van een menu naar het desbetreffende programma-onderdeel te vertakken:

```
 10 PRINT"PROGRAMMAKEUZE:"
 20 PRINT"---------------"
 30 PRINT" 1- REGISTRATIE ARTIKELEN"
 40 PRINT" 2- ARTIKELEN ONDERHOUDEN"
 50 PRINT" 3- REGISTRATIE KLANTEN"
 60 PRINT" 4- KLANTEN VERZORGEN"
 70 PRINT" 5- BESTELLING"
 80 PRINT" 6- REKENING PRINTEN
 90 PRINT" 7- LEVERINGSBON PRINTEN"
100 PROC GET
110 ::::GET X$:IF X$<"1" OR X$>"7" THEN 110
120 Z=VAL(X$)
130 CGOTO Z*10+130
140 CALL REGISRATIE ARTIKELEN
150 CALL ARTIKELEN ONDERHOUDEN
160 CALL REGISTRATIE KLANTEN
```

```
170 CALL VERZORGEN KLANTEN
180 CALL BESTELLING
190 CALL REKENING PRINTEN
200 CALL LEVERINGSBON PRINTEN
```

Dit is een zogenaamde sprong-tabel. Afhankelijk van de menu-keuze, die volgens de GET-opdracht de waarde 1 tot 9 bevat, wordt vertakt naar het desbetreffende subprogramma. Het is slechts een van de vele gebruiksmogelijkheden van CGOTO. U hebt dit vast snel door. Daarna zult u ongetwijfeld eigen toepassingen ontdekken.

5.2.6 Testopdrachten

1) Welke voordelen bieden de symbolische sprongadressen?

   A) De instructies GOTO en GOSUB worden vermeden
   B) Het programma wordt overzichtelijker
   C) Subroutines worden vermeden

2) Welke instructie vervangt de standaard-instructie GOTO?

   A) CALL
   B) EXEC

3) Welke instructie vervangt de standaard-instructie GOSUB?

   A) CALL
   B) EXEC

4) Mag er in een PROC-regel nog een andere instructie staan?

   A) ja
   B) nee hoor

5) Welke instructie beëindigt een met EXEC opgeroepen subroutine?

   A) RETURN
   B) END PROC
   C) PROC

6) Naar welke regel vertakt het programma zich door de volgende CGOTO-instructie?
   10 A=9
   20 CGOTO (A+1)*10+130

   A) regel 1310
   B) regel 210
   C) regel 230
   D) regel 220

6 VARIABELEN

Variabelen nemen in standaard-BASIC altijd een bepaalde waarde aan. Wordt deze waarde veranderd en niet op een bepaalde plaats of met een bepaalde naam in het geheugen opgeslagen, dan kan die oorspronkelijke waarde niet worden teruggehaald. Simon's BASIC biedt u een uitzondering. U kunt met Simon's BASIC de globale waarde van variabelen op een gewenste plaats door een locale waarde vervangen. We hebben dan de beschikking over een waarde, die alleen voor een bepaald programma-onderdeel geldig is. De oorspronkelijke waarde gaat niet verloren, maar kan met een instructie weer worden teruggehaald. Iedere variabele kan dus twee waarden aannemen:

de globale waarde

en

de locale waarde

6.1 LOCAL

Format	LOCAL variabele 1, variabele 2, variabele 3,......
Parameter	variabele - willekeurige variabele (bijvoorbeeld A,A$,A%)
Functie	Het vastleggen van locale variabelen.
Voorbeeld	LOCAL A$,X$,B
	De variabelen A$, XS en B worden als locale variabelen bepaald.

Het aantal variabelen in omvangrijke programma's is meestal zo groot, dat het overzicht gemakkelijk verloren kan gaan. Met de instructie LOCAL kan dit aantal nu sterk worden gereduceerd. U kunt nu in uw programma een enkele variabele op diverse plaatsen gebruiken. Voordat de variabelen veranderd worden, dus op een locale waarde worden gebracht, moeten ze met de instructie LOCAL als locale variabele worden aangegeven. Een eenvoudig voorbeeld:

```
100 REM ******************************
110 REM **  GLOBALE WAARDE VASTLEGGEN **
120 REM ******************************
130 A=120:B%=850:X$="GLOBAAL"
140 PRINT"GLOBALE WAARDE:";A;B%;X$
150 REM ******************************
160 REM **  LOKALE WAARDE VASTLEGGEN  **
170 REM ******************************
180 LOCAL A,B%,X$
190 A=240:B%=1700:X$="LOKAAL"
200 PRINT"LOKALE WAARDE:";A;B%;X$
```

In regel 130 worden de variabelen met de globale waarden gedefinieerd. Regel 180 definieert deze variabelen dan als locale variabelen, waaraan in regel 190 nieuwe waarden zijn gegeven. De instructie LOCAL slaat de globale waarden op, voordat de locale waarden worden ingevoerd. Met de volgende instructie kan de globale waarde weer worden teruggehaald.

6.2 GLOBAL

Format	GLOBAL
Parameter	geen
Functie	De oorspronkelijke waarden weer terughalen.

Deze instructie geeft alle variabelen weer de oorspronkelijke waarde. Om de globale waarde van het vorige programma terug te krijgen, moeten de volgende regels worden toegevoegd:

```
210 REM ******************************
220 REM **  GLOBALE WAARDE HERSTELLEN **
230 REM ******************************
240 GLOBAL
250 PRINT"GLOBALE WAARDE:";A;B%;X$
```

In regel 240 worden dus de oude waarden weer aangenomen. Maar de locale waarden zijn niet verloren gegaan. Met een hernieuwde LOCAL kunnen de variabelen de locale waarden weer aannemen. Voer bijvoorbeeld de instructie 'LOCAL A' in. Let nu op de waarde van de variabelen met 'PRINT A'. U ziet, dat de locale waarde (240) weer werd teruggehaald. Op deze manier kunt u willekeurig tussen globale en locale waarden heen en weer schakelen.

6.3 Testopdrachten

1. Welke taak heeft de instructie LOCAL?

   A) Globale waarden vegen en locale waarden definiëren
   B) Globale waarde opslaan en locale waarde definiëren
   C) Globale waarde opslaan en waarden toekennen aan locale variabelen.
   D) Globale waarden vegen en waarden toekennen aan locale variabelen.

2. Welke taak heeft de instructie GLOBAL?

   A) Locale waarden vegen en globale waarden bepalen.
   B) Locale waarden opslaan en globale waarden terugkrijgen.
   C) Locale waarden opslaan en globale waarden definiëren.
   D) Locale waarden vegen en globale waarden weer terughalen.

3. Welke variabelen kunnen met locale waarden worden gedefinieerd?

   A) Integervariabelen (bijvoorbeeld A%)
   B) Stringvariabelen (bijvoorbeeld S$)
   C) Vlottende komma-variabelen (bijvoorbeeld A)

7 DE BEHANDELING VAN DE GETALLEN

7.1 Rekenkundige operatoren.

7.1.1 MOD

Format	MOD(x,y)
Parameter	x - deeltal
	y - deler
Functie	De rest weergeven bij de deling van een geheel getal
Voorbeeld	PRINT MOD (14,3)
	Toont de rest van de deling (2)

Deze instructie spoort de rest van de deling van een integer getal op. Maar wat zijn eigenlijk integers? Integers zijn gehele getallen, dat wil zeggen getallen zonder waarde achter de komma. De CBM 64 kent alleen integer-getallen van 2 bytes. Dit bespaart veel ruimte in het geheugen. In 2 bytes kunnen we 65535 (2^16 - 1) gehele getallen uitdrukken. Ook ieder adres van het geheugen van de Commodore 64 is een geheel getal van 2 bytes.

De integer-variabelen in BASIC kunnen echter ook negatieve waarden aannemen. Omdat de bit 15 (geheel links) het voorteken bepaalt, blijven er nog maar 15 bits (0 - 14) over voor de uitdrukking van waarden. Een integer-variabele kan dus de waarde -32768 tot +32767 aannemen. Wordt een andere waarde toegekend (bijvoorbeeld A%=33000), dan volgt de foutmelding "ILLEGAL QUANTITY ERROR".

De MOD-instructie accepteert alleen positieve gehele getallen tussen 0 en 65535. Iedere andere waarde veroorzaakt ook hier de foutmelding "ILLEGAL QUANTITY ERROR", dus ook ieder negatief getal. Het volgende programma spoort de rest op van een deling met behulp van de MOD-instructie.

```
10 INPUT"DIVIDEND : ";X
20 INPUT"DELER    : ";Y
30 PRINT"REST     : ";MOD (X,Y)
```

De instructie MOD kan met een variabele parameter binnen een programma en ook direkt worden ingevoerd. U kunt bijvoorbeeld direkt de instructie 'PRINT MOD (144,7)' invoeren. Omdat de integer-variabelen van de Commodore 64 (bijvoorbeeld A%) bij een deling alleen de waarde voor de komma in het geheugen opnemen, moet de rest met de instructie MOD worden gedefinieerd. Een voorbeeld:

```
10 A%=50/12
20 A=50/12
30 PRINT "UITKOMST A%:"A%
40 PRINT "UITKOMST A :"A
```

Na afloop van het programma ziet u het resultaat. De integer-variabele A% heeft van de waarde van de uitkomst van de deling het deel dat voor de komma staat als inhoud. De vlottende komma-variabele A heeft echter het juiste resultaat als inhoud. De instructie MOD maakt nu ook het vaststellen van de rest mogelijk.

```
10 A%=50/12
20 B%=MOD(50,12)
30 PRINT"UITKOMST :"A%"REST"B%
```

Op deze manier kunt u voortaan integer-variabelen door een willekeurig getal delen.

7.1.2 DIV

Format DIV (x,y)

Parameter x - deeltal

y - deler

Functie Het vaststellen van de waarde voor de komma van de divisie.

Voorbeeld PRINT DIV (14,3)

Geeft de waarde voor de komma van het resultaat van de deling.

Met de beide instructies MOD en DIV kunt u zowel het gehele getal van de quotiënt, als de rest van een integerdeling vaststellen. Het volgende programma berekent beide:

```
100 INPUT "DIVIDEND        : ";X
110 INPUT "DELER           : ";Y
120 D=DIV(X,Y):M=MOD(X,Y)
130 PRINT"DE UITKOMST IS"D"REST"M
```

Het gebruik van de beide instructies is dus vrij eenvoudig. Hoe gemakkelijk ze in de praktijk zijn, laat het volgende voorbeeld zien: In paragraaf 7.2 werd een instructie beschreven, waarmee binaire getallen in decimale getallen konden worden omgezet. Maar de verandering van een decimaal getal in een binair getal kan slechts met een programma worden opgelost. Het volgende programma voert deze verandering door met behulp van de instructies MOD en DIV op basis van de restwaarde-methode:

```
100 PROC INVOER
110 ::INPUT"DECIMAAL GETAL (0-65535)";D
120 ::IF D>=0 AND D<=65535 THEN CALL UITVOER
130 ::PRINT CHR$(145);:CALL INVOER
140 PROC UITVOER
150 ::PRINT"BINAIR GETAL:"
160 :: FOR I=1 TO 16
170 ::::R=MOD(D,2):D=DIV(D,2)
180 ::::PRINTTAB(16-I);R;CHR$(145)
190 ::NEXT I
```

Nadat een juist getal werd ingevoerd, wordt uit dit decimale getal een 16-bit binair getal gevormd.

7.1.3 FRAC

Format FRAC(n/x)

Parameter n-getal of uitdrukking (vlottende komma)

Functie Het bepalen van de waarde na de komma bij een reëel getal.

Voorbeeld PRINT FRAC (10/3)

Geeft de waarde na de komma aan bij quotiënten (.333333333).

Wie wel eens van een getal de waarde na de komma met standaard-BASIC heeft vastgesteld, weet dat dit niet zo makkelijk gaat. De integer-waarde van een getal moet namelijk van het reële getal worden afgetrokken. In plaats van de gemakkelijke instructie 'PRINT FRAC (10/3) moet 'PRINT 10/3 - INT(10/3)' worden ingevoerd. Bij de instructies, die tot nu toe werden behandeld, werd een programma voor het omrekenen van decimale getallen in binaire getallen voorgesteld. Nu volgt een programma, waarmee decimale getallen kunnen worden veranderd in hexadecimale.

```
100 DIM A$(15)
110 FOR I= 0 TO 15
120 ::READA$(I)
130 NEXT I
140 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
150 PROC INVOER
160 ::INPUT"DECIMAAL GETAL (0-65535)";D
170 ::IF D>=0 AND D<=65535 THEN CALL UITVOER
180 ::PRINT CHR$(145);: CALL INVOER
190 PROC UITVOER
200 ::PRINT"HEX-GETAL:"
210 ::FOR I= 1 TO 4
220 ::::R=16*FRAC(D/16):D=INT(D/16)
230 ::::PRINTTAB(4-I);A$(R);CHR$(145)
240 ::NEXT I
```

Voer ook hier een getal in tussen 0 en 65535, dat daarna in een hexadecimaal getal van 4 cijfers wordt veranderd.

7.1.4 EXOR

Format	EXOR(n, n1)
Parameter	n,n1 - gerelateerde getallen (0-65535)
Functie	Twee getallen in relatie tot elkaar bewerken met de "exclusive or"-functie.
Voorbeeld	PRINT EXOR (255,1)
	Schakelt bit 0 van de waarde 255 om, dus op nul.

Voor de meesten van u is de Booleaanse algebra onbekend terrein. De Booleaanse uitdrukkingen in standaard-BASIC zijn: OR, AND en NOT. Deze logische operatoren worden bijvoorbeeld in IF-testen gebruikt. Ze worden ook wel relationele operatoren genoemd. Een benaming die misschien duidelijker is, zou 'vergelijkings-operatoren' kunnen zijn. Al deze benamingen geven aan dat de functie van de operator is om een bewerking uit te voeren die berust op de vergelijking van twee getallen met elkaar. Verder kunt u met deze vergelijkings-operatoren een bepaalde plaats in het geheugen manipuleren. De relationele operatoren hebben hier het volgende effect:

OR - bits inschakelen
AND - bits uitschakelen.

Deze instructies zijn nodig voor de programmering van een register, omdat in de Commodore 64 talrijke registers voor de meest uiteenlopende doelen aanwezig zijn (grafiek, geluid). Ieder van deze registers omvat een byte (8 bit). Deze bits worden met behulp van logische vergelijkingen in- en uitgeschakeld. Maar hoe doet u dat nu? Daarvoor moet een te schakelen byte bekend zijn:

	0	0	0	0	0	0	0	0
bit	7	6	5	4	3	2	1	0
waarde	128	64	32	16	8	4	2	1

Een byte bestaat dus uit 8 bits, die genummerd zijn en een bepaalde waarde hebben. Moet nu een van deze bits worden ingeschakeld, dan moet met de waarde van die bit worden vergeleken. Een voorbeeld: Bit 4 van geheugenplaats 53248 moet worden ingeschakeld. De betreffende bewerking zou dan moeten zijn:

```
POKE 53248, PEEK (53248) OR 16
```

De inhoud van de geheugenpositie wordt met de waarde van bit 4 (16) vergeleken. Een ander voorbeeld: De bits 0,1,2 en 3 moeten worden gezet. Dan moeten daarvoor de afzonderlijke waarden worden opgeteld. De betreffende instructie ziet er dan zo uit:

```
POKE 53248, PEEK (53248) OR 15
```

Het is moeilijker om de bits uit te schakelen, omdat daarvoor niet met de waarde van de gezette bits van een vergelijkingsbyte door een AND-functie moet worden vergeleken. In plaats daarvan met 255 minus de bitwaarde. Moet de bit 0 worden uitgeschakeld, dus op 0 worden gezet, dan moet er met 255 - 1(254) worden vergeleken. De betreffende instructie voor het uitschakelen van de ingeschakelde bits, ziet er als volgt uit:

```
POKE 53248, PEEK (53248) AND (255-16)
POKE 53248, PEEK (53248) AND (255-15)
```

of zo:

```
POKE 53248, PEEK (53248) AND (255-31)
```

Simon's BASIC biedt nu een aanvulling: EXOR. Met EXOR worden de bits omgeschakeld. Dat wil zeggen: wanneer een bit voor de EXOR-operatie was gezet, wordt het gewist en omgekeerd. Een voorbeeld: Bit 7 van geheugenplaats 53250 moet worden omgeschakeld. De desbetreffende instructie daarvoor is:

```
POKE 53250, EXOR (PEEK(53250),128)
```

De schrijfwijze is iets veranderd, omdat het format van EXOR niet is, zoals bij OR en AND 'n EXOR n1', maar 'EXOR(n,n1)'. Misschien had Simon's BASIC hier moeten worden aangepast.

7.1.5 Testopdrachten

1. Welke uitkomst geeft de instructie PRINT MOD (33,7)?

A) 4
B) 5
C) 0
D) 28

2. Welke uitkomst geeft de instructie PRINT DIV (33,7)?

   A) 4
   B) 5
   C) 0
   D) 28

3. Welke maximum-waarden mogen de MOD- en DIV-parameter niet overschrijden?

   A) 32767
   B) 65535
   C) 1e+38

4. Welke waarde wordt door de instructie FRAC verkregen?

   A) de waarde na de komma van een integer getal
   B) de waarde voor de komma van een integer getal
   C) de waarde na de komma van een getal met vlottende komma
   D) de waarde voor de komma van een getal met vlottende komma

5. De bits 0 en 1 van de geheugenplaats 53300 moeten worden omgeschakeld. Welke instructie is juist?

   A) POKE 53300, PEEK (53300) EXOR 3
   B) POKE 53300, EXOR (PEEK (53300),3)

7.2 De omzetting van getallen

7.2.1 %

Format	% binaire waarde
Parameter	binaire waarde - binair getal van acht bits
Functie	binair - decimaal omrekenen
Voorbeeld	PRINT %11111111
	geeft de decimale waarde 255 aan.

Met het teken '%' worden binaire getallen van 8 bits aangeduid, die dan kunnen worden weergegeven of opgeslagen. Het binaire getal van 8 bits mag alleen de cijfers 0 en 1 bevatten, omdat anders de foutmelding "NOT BINARY CHAR" wordt gegeven. Niet alleen de schrijfwijze

```
PRINT %00001111
```

is mogelijk, maar ook

```
POKE 53280,%000111000
```

of

```
A+%00001110
```

Het binaire getal wordt telkens in een decimaal getal omgezet. Bij de logische vergelijking biedt deze schrijfwijze belangrijke voordelen. Om enkele bits gericht te sturen, hoeft de vergelijking niet door de programmeur te worden omgerekend. De instructie voor het inschakelen van de bits 0 en 2 van de geheugenplaats 49152 kan daardoor eenvoudiger geschreven worden:

```
POKE 49152,PEEK(49152) OR %00000101
```

De %-schrijfwijze vereenvoudigt het werken met de bits en bytes. In paragraaf 7.1.2 vindt u een programma, waarmee decimale getallen kunnen worden omgezet in 16-bits-binaire getallen.

7.2.2 $

Format	$ hex-waarde
Parameter	hex-waarde - hexadecimaal getal met 4 digits

Functie omzetting hexadecimaal - decimaal

Voorbeeld PRINT $00FF

geeft de decimale waarde 255

Hier wordt een hexadecimaal getal van 4 digits in een decimaal getal omgezet. Wanneer niet een getal van 4 digits wordt aangegeven met geldige hexadecimaal-aanduidingen wordt de foutmelding "NOT HEX CHAR" aangegeven. Deze schrijfwijze kan heel handig zijn. Ook BASIC-programmeurs gebruiken routines van het besturingssysteem. Interessante routines haalt u uit de ROM-listing van de Commodore 64, die in het boek "64 INTERN" van DATA BECKER/NEDERLANDS* aanwezig is. De adressen van deze routines zijn meestal hexadecimaal aangegeven. Wilt u nu een routine gebruiken, dan moet het hexadecimale adres eerst naar decimaal worden omgerekend. Dat extra werk valt weg met het gebruik van het $-teken. Wilt u bijvoorbeeld het bedrijfssysteem-adres $FCE 2 (koude start) aanspreken, voer dan alleen de volgende instructie in:

```
PRINT $FCE2
```

De computer geeft dan het decimale adres van de routine aan (64738). Met SYS 64738 kan deze routine worden gestart, die de computer in de toestand als direkt na het inschakelen brengt. Maar het is eenvoudiger om de routine op te roepen met de instructie:

```
SYS $FCE2
```

Het $-teken kan dus overal worden ingevoerd met hexadecimale waarden, waar anders decimale waarden werden ingevoerd.

7.2.3 Testopdrachten

1. Welke van de volgende instructies leidt tot een foutmelding?

   A) PRINT %10101000
   B) POKE 53000,EXOR (PEEK(53000),%00000011)
   C) PRINT $0011

2. Waarop moet u letten bij de invoer van het teken %?

A) Het binaire getal mag maximaal 8 cijfers hebben.
B) Het binaire getal mag maximaal 16 cijfers hebben.
C) Uitsluitend de binaire getallen 0 en 1 mogen worden gebruikt.
D) Alleen het binaire cijfer 1 mag worden gebruikt.

3. Welke van de volgende instructies leidt tot een foutmelding?

A) PRINT $FF
B) POKE 1024, $0000
C) A%=$00FF+$000F

4. Waarop moet u letten bij de invoer van het kenmerk $?

A) Het hex-getal mag maximaal 2 cijfers hebben.
B) Het hex-getal mag maximaal 4 cijfers hebben.
C) Er mogen alleen decimale cijfers (0-9) worden gebruikt.
D) Er mogen alleen hexadecimale cijfers (0-9 en A-F) worden gebruikt.

8 STRINGOPERATIES

8.1 INSERT

Format	INSERT ("string 1", "string 2", pos)
Parameter	string 1 - in te voeren string
	string 2 - resulterende string
	pos - positie, volgens welke moet worden ingevoerd
Functie	Het invoeren van een string in een andere string.
Voorbeeld	10 X$=INSERT ("CBM V2", "EXPANDED BASIC",8)
	20 PRINT X$
	Voegt de eerste string in vanaf de achtste plaats in de tweede string; "EXPANDED CBM V2 BASIC" wordt aangegeven.

Met deze instructie kunt u een string in een andere string invoeren op een willekeurige plaats. Naast string-constanten kunnen ook stringvariabelen worden aangegeven. Zo kunnen de bovengenoemde BASIC-regels ook als volgt worden veranderd:

```
10 A$=" CBM V2"
20 B$="EXPANDED BASIC"
30 X$=INSERT (A$,B$,8)
40 PRINT X$
```

Twee stringvariabelen werden hier gebruikt. Maar ook de derde parameter, de positie, kan als variabele worden aangegeven. Daarmee zijn nu alle 3 parameters van INSERT als variabelen te gebruiken:

```
10 A$=" CBM V2"
20 B$="EXPANDED BASIC"
30 C=8
40 X$=INSERT (A$,B$,C)
```

```
50 PRINT X$
```

Bij het invoeren van de instructie INSERT moet u letten op de volgende punten:

1. De in te voeren string mag niet groter zijn dan de omvattende string.
2. De resulterende string mag de 255 ~~regels~~ tekens niet overschrijden.
3. De positie-weergave mag hoogstens de lengte hebben van de omvattende string min 1. Er mag dus niet toegevoegd worden, maar alleen ingevoegd.
4. Wanneer de instructie in de direkt-modus wordt ingevoerd (niet in een programmaregel), mogen alleen string-variabelen, geen tekenreeksen (bijvoorbeeld "XYZ") worden gebruikt.

Als u de punten 1 en 3 negeert, wordt de foutmelding "INSERT TOO LARGE" gegeven. Omdat een string in geen geval meer dan 255 regels mag omvatten, zal bij negeren van punt 2 de foutmelding "STRING TOO LARGE" verschijnen. Punt 4 lijkt vast onwaarschijnlijk. De invoer van INSERT in de direkt-modus is weliswaar niet zonder meer zinvol, maar de reactie van Simon's BASIC geeft te denken. Na afloop van het laatstgenoemde programma blijven de stringvariabelen A$ en B$ behouden, zodat de volgende instructie in de direkt-modus een goed resultaat heeft:

```
PRINT INSERT (A$,B$,8)
```

Er wordt een correcte string "EXPANDED CBM V2 BASIC" gegeven. Ook wanneer de eerste parameter tegen een string-constante wordt uitgewisseld, is het resultaat juist:

```
PRINT INSERT (" CBM V2", B$,8)
```

Maar nu opgelet: wanneer ook de tweede parameter als stringconstante wordt aangegeven, heeft dit verbazingwekkende gevolgen:

```
PRINT INSERT (" CBM V2", "EXPANDED BASIC",8)
```

Op het beeldscherm verschijnt een string, die helemaal in de war is geraakt:

```
EXPANDEDD BASIC BASIC
```

Maar deze instructie wordt bijna nooit in de direkt-modus ingevoerd. Dus dit mankement in Simon's BASIC kan buiten beschouwing worden

gelaten. In een programma werkt de instructie probleemloos.

8.2 INST

Format	INST ("string 1", "string 2", pos)
Parameter	string1 - de getransporteerde string
	string2 - de oude string
	pos - positie, waarop moet worden begonnen met overschrijven
Functie	Het overschrijven van een string met een andere.
Voorbeeld	10 X$=INST ("130","HOEVEELHEID 500 STUKS",12)
	20 PRINT X$
	Overschrijft de tweede string met de eerste vanaf de positie 12; de string "HOEVEELHEID 130 STUKS" wordt uitgevoerd.

De parameters van deze instructies zijn gelijk aan die van de voorheen beschreven INSERT-instructies. De instructie INST voegt echter niet in, maar overschrijft de tweede string. De eerste string mag groter zijn dan de tweede. Dat heeft echter vervelende gevolgen, wanneer de tweede parameter een niet gedefinieerde stringvariabele is. Dan geeft Simon's BASIC het op en laat geen stringbewerking meer toe. De positionering mag de tweede string niet overschrijden. Wanneer de tweede string bijvoorbeeld 20 tekens omvat, mag de positie-opgave niet groter zijn dan 20. Wordt hiermee geen rekening gehouden, dan wordt geen foutmelding gegeven, maar is de ontstane string niet naar wens opgebouwd. Natuurlijk mag de nieuwe string ook niet meer dan 255 tekens bevatten. Een interessant toepassingsgebied vindt de INST-instructie bijvoorbeeld bij de opbouw van een record voor relatieve opslag in het geheugen. Een record bestaat meestal uit verscheidene velden, die zich altijd op een bepaalde positie moeten bevinden. Een record van een adressenbestand kan bijvoorbeeld als volgt zijn opgebouwd:

Veld	Positie in record	Variabele
Naam	0	N$
Voornaam	25	V$
Straat	45	S$
Plaats	75	P$

Deze vier velden moeten nu worden veranderd in een string (R$), waarvan de opbouw moet overeenkomen met de bovengenoemde structuur. De string R$ moet 100 tekens bevatten en van te voren met spaties worden bepaald. Het volgende fragment van het programma geeft aan hoe het record in de string R$ wordt opgebouwd:

```
100 R$=DUP(" ",100)
110 R$=INST(N$,R$,0)
120 R$=INST(V$,R$,25)
130 R$=INST(S$,R$,45)
140 R$=INST(P$,R$,75)
```

De string R$, die 100 spaties bevat, wordt in regel 100 met de instructie DUP voortgebracht (zie paragraaf 8.4).

De volgende vier INST-instructies zetten 4 variabelen op de desbetreffende plaats in de string R$. Het totale, relatief opgebouwde adressenbestand is dan uniform gestructureerd, zodat na het inlezen van een record ieder willekeurig veld op de goede plaats gevonden wordt.

Overigens willen we alle lezers, die meer over de relatieve opslag van gegevens met de diskdrive VC-1541 willen weten, adviseren "HET GROTE FLOPPYBOEK" van DATA BECKER/NEDERLANDS* te lezen.

Verder is de INST-instructie erg geschikt voor het uitvoeren van tabellen met de printer. De afzonderlijke velden van een printregel worden voor het afdrukken op de betreffende posities van een string geplaatst. Na het geschikt maken voor de verwerking wordt de string dan geprint. In principe is dit vergelijkbaar met de eerder beschreven bewerking van gegevens voor een bestand. Wie deze stringbewerking ooit met standaard BASIC heeft opgelost, weet dat de INST-instructie deze bewerking veel eenvoudiger maakt.

8.3 PLACE

Format	PLACE
Parameter	string 1 - de gezochte string
	string v - de stringvariabele, waarin gezocht moet worden
Functie	Het zoeken van een tekenreeks in een string.
Voorbeeld	A$="123*4567"
	PRINT PLACE ("*",A$)
	Geeft de positie aan van de ster in de string A$

Met PLACE beschikt u over een zeer effectieve instructie, die met standaard BASIC erg moeilijk te vervangen is. De instructie PLACE kan ook goed met INST worden gecombineerd. Een voorbeeld: de decimaal-punt is zowel bij de Commodore 64 als bij alle andere computers volgens de Amerikaanse schrijfwijze een punt. Maar in Nederland schrijven we een komma. Met PLACE kunnen nu bedragen op de Nederlandse manier worden geschreven. Kijk eerst eens naar het volgende programma:

```
10 INPUT"BEDRAG: ";B
20 B$=STR$(B)
30 P=PLACE(".",B$)
40 B$=INST(",",B$,P-1)
50 PRINT"BEDRAG: ";B$
```

In regel 10 wordt een bedrag ingelezen en in een numerieke variabele opgeslagen. Regel 20 verandert de numerieke variabele in een stringvariabele. Nu wordt de plaats van de punt in regel 30 vastgesteld en in de variabele P opgeslagen. Daarna wordt in deze regel de punt, waarvan de positie is bewaard in P-1, vervangen door een komma. Tenslotte wordt dan het bedrag op de Nederlandse manier aangegeven. De positie in de INST-instructie moet met 1 worden verminderd, omdat de instructies INSERT en INST bij het tellen beginnen bij 0 (nul). Positie 1 van de PLACE-instructie is bijvoorbeeld de positie 0 bij INSERT of INST. De oplossing van dit probleem in standaard BASIC zou ongeveer drie keer zoveel regels in beslag nemen. Daar wordt een programma natuurlijk niet duidelijker

van.

Niet alleen een teken, maar ook een tekenreeks kan met de PLACE-instructie worden opgespoord. Het volgende voorbeeld maakt dit duidelijk:

```
10 A$="DATA BECKER/NEDERLANDS* TRAINT DE GEEST"
20 PRINT PLACE ("* T",A$)
```

Voer dit programma in en start het met RUN. Op het beeldscherm wordt 23 getoond, wat met de positie van de tekenreeks "* T" overeenkomt. Deze handige instructie interesseerde ons zo, dat we een programma ontwikkelden voor de opslag en opsporing van memo's. Ieder memo bestaat uit maximaal 2 regels op het beeldscherm en wordt op diskette opgeslagen. Het totale memobestand wordt als tabel in de computer geladen, wat samen met de instructie PLACE een snelle opsporing van trefwoorden mogelijk maakt. De listing van dit programma:

```
1000 REM ********************
1010 REM     VOORBEREIDINGEN
1020 REM ********************
1030 POKE 53280,2:POKE53281,2
1040 PRINTCHR$(158)
1050 DIM I$(4000)
1060 PROC MENU
1070 ::REM ********************
1080 ::REM          MENU
1090 ::REM ********************
1100 ::EXEC PROGRAMMAKOP
1110 ::PRINT" KIES DE GEWENSTE FUNCTIE:"
1120 ::PRINT" ";DUP("-",37):PRINT
1130 ::PRINTTAB(8);" -1- MEMOBESTAND LADEN"
1140 ::PRINTTAB(8);" -2- MEMOBESTAND SAVEN"
1150 ::PRINTTAB(8);" -3- MEMO INVOEREN"
1160 ::PRINTTAB(8);" -4- MEMO WISSEN"
1170 ::PRINTTAB(8);" -5- MEMO ZOEKEN"
1180 ::PRINTTAB(8);" -6- PROGRAMMA-EINDE"
1190 ::LOOP
1200 ::::GETX$
1210 ::::X=VAL(X$)
1220 ::::EXIT IF X>0 AND X<7
1230 ::END LOOP
1240 ::IF X=1THEN EXEC MEMOBESTAND LADEN
1250 ::IF X=2THEN EXEC MEMOBESTAND SAVEN
1260 ::IF X=3THEN EXEC MEMO INVOEREN
```

```
1270 ::IF X=4THEN EXEC MEMO WISSEN
1280 ::IF X=5THEN EXEC MEMO ZOEKEN
1290 ::IF X=6THEN EXEC PROGRAMMA-EINDE
1300 ::CALL MENU
1310 PROC MEMOBESTAND LADEN
1320 ::REM ********************
1330 ::REM   MEMOBESTAND LADEN
1340 ::REM ********************
1350 ::EXEC PROGRAMMAKOP
1360 ::INPUT"NAAM VAN BESTAND:";DN$
1370 ::OPEN1,8,2,DN$:CLOSE1
1380 ::OPEN15,8,15
1390 ::INPUT#15,F1$,F2$,F3$,F4$
1400 ::IF F1$="00"THEN CLOSE 15:CALL LEZEN
1410 ::::PRINT:PRINT"DISKETTEFOUT!"
1420 ::::PRINT"----------------"
1430 ::::PRINTF1$","F2$","F3$","F4$
1440 ::::CLOSE15
1450 ::::CALL TERUGSPRONG1
1460 ::PROC LEZEN
1470 ::::OPEN1,8,2,DN$
1480 ::::PRINT"NIEUW- OF BIJLADEN (N/B)?"
1490 ::::REPEAT
1500 ::::::GET X$
1510 ::::UNTIL X$="N" OR X$="B"
1520 ::::IF X$="N" THEN Y=0
1530 ::::LOOP
1540 ::::::Y=Y+1
1550 ::::::INPUT#1,I$(Y)
1560 ::::::EXIT IF ST=64
1570 ::::END LOOP
1580 ::::CLOSE1
1590 ::::PRINT"DE COMPUTER BEVAT MOMENTEEL"Y"GEGEVENS"
1600 ::PROC TERUGSPRONG1
1610 ::::PRINT:PRINT">>RETURN<<"
1620 ::::REPEAT
1630 ::::::GETX$
1640 ::::UNTIL X$=CHR$(13)
1650 END PROC
1660 PROC MEMOBESTAND SAVEN
1670 ::REM ********************
1680 ::REM   MEMOBESTAND SAVEN
1690 ::REM ********************
1700 ::EXEC PROGRAMMKAKOP
1710 ::IF Y=0THENPRINT"GEEN DATA!!":CALL TERUGSPRONG2
1720 ::INPUT"BESTANDSNAAM :";DN$
1730 ::PRINT"BESTAND "DN$" WORDT GESAVED!"
1740 ::OPEN 1,8,2,CHR$(64)+":"+DN$+",S,W"
1750 ::FOR I=1TOY
```

```
1760 ::::PRINT#1,I$(I)
1770 ::NEXT I
1780 ::CLOSE 1
1790 ::PROC TERUGSPRONG2
1800 ::::PRINT:PRINT">>RETURN<<"
1810 ::::REPEAT
1820 ::::::GET X$
1830 ::::UNTIL X$=CHR$(13)
1840 END PROC
1850 PROC MEMO INVOEREN
1860 ::REM ****************
1870 ::REM   MEMO INVOEREN
1880 ::REM ****************
1890 ::EXEC PROGRAMMAKOP
1900 ::IF Y>0 THEN CALL INVOER
1910 ::::INPUT"BESTANDSNAAM: ";DN$
1920 ::PROC INVOER
1930 ::EXEC PROGRAMMAKOP
1940 ::::Y=Y+1
1950 ::::PRINT"VOERT U DE INFORMATIE NR."Y"IN:"
1960 ::::PRINT"(MAXIMAAL 2 REGELS)"
1970 ::::PRINTDUP("-",39)
1980 ::::INPUT I$(Y)
1990 ::::PRINTDUP("-",40);
2000 ::::PRINT"GOED (J/N)?"
2010 ::::REPEAT
2020 ::::::GETX$
2030 ::::UNTIL X$="J" OR X$="N"
2040 ::::IF X$="N" THEN Y=Y-1:CALL INVOER
2050 ::::PRINT"NOG MEER INVOEREN (J/N)?"
2060 ::::REPEAT
2070 ::::::GET X$
2080 ::::UNTIL X$="J" OR X$="N"
2090 ::::IF X$="J"THEN CALL INVOER
2100 END PROC
2110 PROC MEMO WISSEN
2120 ::REM **************
2130 ::REM   MEMO WISSEN
2140 ::REM **************
2150 ::EXEC PROGRAMMAKOP
2160 ::IFY=OTHENPRINT"GEEN DATA!!":CALL TERUGSPRONG3
2170 ::INPUT"NUMMER VAN INFORMATIE: ";N
2180 ::PRINTDUP("-",39)
2190 ::PRINT"  ";I$(N)
2200 ::PRINTDUP("-",39)
2210 ::PRINT"WISSEN (J/N)?"
2220 ::REPEAT
2230 ::::GET X$
2240 ::UNTIL X$="J" OR X$="N"
```

```
2250 ::IF X$="N" THEN CALL TERUGSPRONG3
2260 ::FORI=N TO Y
2270 ::::I$(I)=I$(I+1):
2280 ::NEXT I
2290 ::Y=Y-1
2300 ::PRINT"INFORMATIE IS GEWIST!"
2310 ::PROC TERUGSPRONG3
2320 ::::PRINT">>RETURN<<"
2330 ::::REPEAT
2340 ::::::GET X$
2350 ::::UNTIL X$=CHR$(13)
2360 END PROC
2370 PROC INFORMATIE ZOEKEN
2380 ::REM ********************
2390 ::REM   INFORMATIE  ZOEKEN
2400 ::REM ********************
2410 ::EXEC PROGRAMMAKOP
2420 ::IFY=0THENPRINT"GEEN GEGEVENS !!":CALL TERUGSPRONG4
2430 ::PRINT"HOEVEEL TREFWOORDEN (1-5)?"
2440 ::REPEAT
2450 ::::GET X$
2460 ::UNTIL X$>"0" AND X$<"6"
2470 ::S=VAL(X$):VK$="O"
2480 ::EXEC PROGRAMMAKOP
2490 ::IFS=1THENCALL VERDER
2500 ::PRINT"EN/OF VERBINDING (E/O)?"
2510 ::REPEAT
2520 ::::GET X$
2530 ::UNTIL X$="E" OR X$="O"
2540 ::VK$=X$
2550 ::PROC VERDER
2560 ::PRINT"BEELDSCHERM OF PRINTER (B/P)?"
2570 ::REPEAT
2580 ::::GET X$
2590 ::UNTIL X$="B" OR X$="P"
2600 ::IF X$="B"THEN OPEN1,3:ELSE:OPEN1,4
2610 ::AU$=X$:PRINTDUP("-",20)
2620 ::FOR I=1TOS
2630 ::::PRINT"TREFWOORD"I": ";
2640 ::::INPUTS$(I)
2650 ::NEXTI
2660 ::FOR S1=1TOY
2670 ::::U1=0:FL=0
2680 ::::FORS2=1 TO S
2690 ::::::Z=PLACE (S$(S2),I$(S1))
2700 ::::::IF Z<>0 AND VK$="O"THEN FL=1:EXEC UITVOER
2710 ::::::IF Z=0 AND VK$="E"THEN CALL NEXT S1
2720 ::::::IF Z<>0 AND VK$="E"THENU1=U1+1
2730 ::::::IFFL=1THENCALL NEXT S1
```

```
2740 ::::NEXT S2
2750 ::::IF U1=S THEN EXEC UITVOER
2760 ::PROC NEXT S1
2770 ::NEXT S1
2780 ::PRINT"ZOEKEN BEEINDIGD!"
2790 ::CALL TERUGSPRONG4
2800 ::PROC UITVOER
2810 ::::IF AU$="B"THEN EXEC PROGRAMMAKOP
2820 ::::PRINT#1,"INFO NR.";S1
2830 ::::PRINT#1,DUP("-",39)
2840 ::::PRINT#1,I$(S1)
2850 ::::PRINT#1,DUP("-",39)
2860 ::::PRINT#1
2870 ::::IF AU$="P"THEN CALL TERUG 1"
2880 ::::::PRINT ">>RETURN<<"
2890 ::::::REPEAT
2900 ::::::::GET X$
2910 ::::::UNTIL X$=CHR$(13)
2920 ::::PROC TERUG1
2930 ::END PROC
2940 ::PROC TERUGSPRONG4
2950 ::::PRINT">>RETURN<<"
2960 ::::REPEAT
2970 ::::::GETX$
2980 ::::UNTIL X$=CHR$(13)
2990 ::::::CLOSE1
3000 END PROC
3010 PROC PROGRAMMA-EINDE
3020 ::REM ******************
3030 ::REM   PROGRAMMA-EINDE
3040 ::REM ******************
3050 ::EXEC PROGRAMMAKOP
3060 ::PRINT"WEET U HET ZEKER?(J/N)"
3070 ::REPEAT
3080 ::::GET X$
3090 ::UNTIL X$="J" OR X$="N"
3100 ::IF X$="N" THEN END PROC
3110 ::EXEC PROGRAMMAKOP
3120 ::PRINT" HET PROGRAMMA KAN MET DE INSTRUCTIE":PRINT
3130 ::PRINT"               'CALL MENU'":PRINT
3140 ::PRINT"   WEER WORDEN GESTART, ZONDER DAT":PRINT
3150 ::PRINT"       GEGEVENS VERLOREN GAAN!!"
3160 ::END
3170 PROC PROGRAMMAKOP
3180 :: REM *****************
3190 :: REM    PROGRAMMAKOP
3200 :: REM *****************
3210 :: PRINTCHR$(147);
3220 :: PRINTDUP("=",40);
```

```
3230 :: PRINTCHR$(18);"    ONGEFORMATEERD INFORMATIE-";
3235 :: PRINT"SYSTEEM    ";CHR$(146);
3240 :: PRINTDUP("=",40);
3250 PRINT"GEGEVENS:"Y"    BESTANDSNAAM: "DN$
3260 ::PRINTDUP("=",40);
3270 ::PRINT:PRINT
3280 END PROC
```

Het is echt de moeite waard om dit programma in te voeren. U hebt daarmee een informatiesysteem, dat u op allerlei manieren kunt gebruiken. Na het starten meldt het programma zich met een menu, dat 6 functies heeft:

- 1 - MEMOBESTAND LADEN

Na de keuze van dit menu wordt gevraagd naar de naam van het bestand. U hebt uw bestand natuurlijk een naam gegeven waaronder het is opgeslagen. Wilt u opnieuw laden of alleen bijladen? Als u al een bestand had geladen, kunt u andere memo's toevoegen. Een nieuw samengesteld bestand kan ook onder een andere naam worden opgeslagen in het geheugen. Wanneer het programma de records, die in het geheugen aanwezig zijn, heeft getoond, komt u door middel van de RETURN weer terug in het menu.

- 2 - MEMOBESTAND OPSLAAN IN HET GEHEUGEN

Deze subroutine vermeldt "GEEN GEGEVENS!!", als hij wordt opgeroepen, wanneer in de computer geen gegevens zijn opgeslagen. Wanneer u de bestandsnamen aangeeft, wordt het bestand op diskette vastgelegd. Voorzichtigheid is hier echter geboden: een bestand, dat eventueel onder deze naam reeds aanwezig is, wordt overschreven! Ook hier komt u door middel van de RETURN-toets weer terug in het menu.

- 3 - MEMO'S INVOEREN

Deze functie wordt gekozen wanneer u voor het eerst gegevens wilt registreren, of wanneer u het bestand, dat reeds is opgeslagen, wilt uitbreiden. Wanneer er geen informatie is opgeslagen, vraagt het programma naar de naam van het bestand, die dan net als het aantal memo's in de computer in de kopregel wordt aangegeven. Is deze naam ingevoerd, dan verschijnt het invoermasker op het beeldscherm. Nu kunt u een record invoeren dat maximaal 2

beeldschermregels mag omvatten. Daarbij wordt gevraagd of de input juist is. Voert u hier de 'N' voor nee in, dan kunt u de input herhalen. Bij 'J' vraagt het programma of u nog meer wilt registreren. Zo nee ('N'), dan verschijnt het menu weer.

- 4 - MEMO'S UITWISSEN

Natuurlijk zijn veel gegevens op den duur niet meer nodig. Termijnen bijvoorbeeld, die reeds verstreken zijn. In dit onderprogramma kunt u nu gericht gegevens wissen. Daarvoor moet u het nummer van de memo's invoeren. Is het nummer onbekend, dan moet dit met behulp van optie 6 van het menu worden bepaald. Voordat er definitief wordt besloten, of het record gewist kan worden, verschijnt het hele record nog een keer op het beeldscherm. Wanneer de toets 'N' wordt bediend, dient het menu zich weer aan. Let erop, dat na het wissen van een record het bestand in de computer opnieuw georganiseerd wordt. Alle volgende records schuiven dan een plaats op.

- 5 - MEMO'S ZOEKEN

Dit is de kern van het programma. U hebt weinig aan het omvangrijkste bestand, wanneer er niet binnen de kortste keren belangrijke informatie kan worden opgevraagd. In dit programma-onderdeel bestaat de mogelijkheid het bestand gemakkelijk te overzien en een overzicht te maken op printer of beeldscherm. Na de keuze moet het aantal trefwoorden (1-5) waarnaar gezocht moet worden in ieder record, worden aangegeven. Wanneer er meer trefwoorden worden gekozen, volgt een belangrijke vraag. Wilt u door middel van EN of door middel van OF vergelijken? Veronderstel, dat u naar twee trefwoorden wilt zoeken. Er bestaan dan twee mogelijkheden voor het opsporen van beide trefwoorden. Moet de gezochte zin uit het bestand beide trefwoorden bevatten, dan moet er door middel van EN worden vergeleken en moet dus de 'E'-toets worden ingedrukt. Wanneer het record een van de beide trefwoorden moet hebben ("...of..."), dan moet er een vergelijking door 'O' worden gemaakt. Dat is dus niet zo moeilijk. U kunt zo maximaal vijf trefwoorden invoeren. Na de vraag met betrekking tot de vergelijking bepaalt u of de gevonden memo's op de printer of op het beeldscherm moeten worden uitgevoerd. Daarna voert u de trefwoorden in.

Wanneer het programma aan het eind van het bestand is gekomen, wordt dit vermeld. Door de RETURN komt u weer terug in het menu.

- 6 - HET EINDE VAN HET PROGRAMMA

Een degelijk ontwikkeld programma heeft natuurlijk ook een behoorlijk einde. De nooduitgang 'RUN/STOP-toets' is hier niet op zijn plaats. Wanneer u deze functie kiest, wordt er nog eens gevraagd, of het programma ook werkelijk moet worden beëindigd. Bij 'N' verschijnt het menu en bij 'J' verschijnt de mededeling, hoe u het programma nogmaals kunt starten zonder uw gegevens verloren te laten gaan.

Maak gebruik van het geheugen van de CBM 64 en vul dit met uw persoonlijke databank.

8.4 DUP

Format DUP ("string",n)

Parameter string - string of stringvariabele

n - dupliceerfactor

Functie het kopiëren van een tekenreeks

Voorbeeld PRINT DUP("-",40)

Vult een beeldschermregel met min-teken

Deze instructie heeft u vast al ontdekt in het laatste programma. Met DUP vervallen instructies als

'PRINT"--"'

Niet alleen enkele tekens kunnen worden gekopieerd, ook langere tekenreeksen kunnen willekeurig worden vermenigvuldigd. Een voorbeeld:

```
10 A$=DUP ("SOS ",10).
20 PRINT A$
```

De beide parameters zijn variabel. De volgorde van instructies bevestigt dit.

```
10 A$="SOS "
20 FOR I=1 TO 10
30 PRINT DUP(A$,I)
40 NEXT
```

Ook controletekens, zoals cursorbesturingstekens, kunnen worden gekopieerd. Het volgende programma-onderdeel gebruikt de ASCII-waarde van 'CURSOR UP' (CHR$ (145)):

```
10 PRINT"REGEL 10"
20 PRINT DUP(CHR$(145),10)
30 PRINT"REGEL 30"
```

Nu vervalt dus de opbouw van een lus, om de cursor over een aantal plaatsen te bewegen. Natuurlijk kan met deze instructie ook een regel worden onderstreept bij gebruik van de printer:

```
10 OPEN 1,4
20 PRINT#1,DUP("-",80)
30 CLOSE 1
```

DUP is een instructie, waarop velen lang hebben gewacht.

8.5 Testopdrachten

1. Welk resultaat leveren de volgende regels op?
   10 A$="AAA BBB CCC"
   20 PRINT INSERT (" XXX",A$,4)

A) AAAXXX BBB CCC
B) AAA XXX BBB CCC
C) AAA XXXBBB CCC

2. Waarop moet u letten bij de invoer van de instructie INSERT?

A) De in te voeren string moet kleiner zijn dan de opnemende string.
B) De in te voeren string moet groter zijn dan de opnemende string.
C) De resulterende string mag de 255 tekens niet overschrijden.

3. Welke van de volgende INSERT-instructies veroorzaken deze volgorde van tekens?
KLAGENFURT

A) 10 PRINT INSERT ("LAGENF","KURT",1)
B) 10 PRINT INSERT ("LAGENF","KURT",0)
C) 10 PRINT INSERT ("LAGENF","KURT",2)
D) geen enkele van deze instructies.

4. Wat is het resultaat van de volgende regels?
10 A$='AAABBBCCCDDD"
20 PRINT INST (" ",A$,4)

A) AAA BBBCCCDDD
B) AAAB BBCCCDDD
C) AAAB BCCCDDD
D) AAA BBCCCDDD

5. Welke positie wordt er met de volgende instructies aangegeven?
10 A$="AA BB CC"
20 PRINT PLACE (" ",A$)

A) 1
B) 2
C) 3

6. Welke van de volgende instructies veroorzaakt niet de waarde 12?

A) 10 PRINT PLACE (" ","AAA.BBB.CCC DDD.EEE")
B) 10 PRINT PLACE ("XY","YX X Y XX YXY XX")

C) 10 PRINT PLACE ("100","1000100010100100")

7. Welke van de volgende instructies geeft deze tekenreeks niet: "------------"?

   A) PRINT DUP ("-",12)
   B) PRINT DUP "--",4)
   C) PRINT DUP "----,3)

8. Kan de instructie DUP ook richttekens (bijvoorbeeld CURSOR HOME) dupliceren?

   A) ja
   B) nee

9. Hoe lang mogen tekenreeksen, die met DUP zijn geproduceerd, hoogstens zijn?

   A) 256 tekens
   B) 1000 tekens
   C) 255 tekens
   D) 1024 tekens

9 CONTROLE VAN DE OUTPUT

Simon's BASIC heeft de mogelijkheid om de output vrij gemakkelijk op het beeldscherm vorm te geven. Met de volgende instructies beschikt u over de hulpmiddelen om de uiterlijke vorm van uw programma tot professioneel niveau te verfijnen. Zo kunt u de output formateren en laten knipperen, of heel gemakkelijk complete beeldschermgebieden behandelen. Dit verleent een optimaal gemak en een hoge snelheid aan uw programma. Eerst worden de formatinstructies, dan de knipper-, "vul-" en tenslotte de scroll-instructies besproken. Om de instructie snel te leren kennen, kunt u ze het beste allemaal direkt uitproberen.

9.1 Geformateerde uitvoer

Onder geformateerde uitvoer verstaan we de geordende positionering van de commentaren, berekeningen en dergelijke, die volgens bepaalde criteria op het beeldscherm zichtbaar moeten worden gemaakt. Het originele CBM-BASIC biedt enkele mogelijkheden om inhouden van het beeldscherm geordend weer te geven. Alle instructies, die voor dit doel in CMB-BASIC werden opgenomen, kunnen in samenhang met de PRINT-instructie worden gebruikt (behalve POS; vergelijk eventueel met het Handboek).

Omdat de programmeur vaak niet zo goed kan omgaan met deze commando's, worden hun functies en mogelijkheden hier kort weergegeven. In dit verband moet eerst een belangrijk begrip worden verklaard:

De cursor

De cursor geeft die positie op het beeldscherm aan, waar het eerstvolgende teken wordt geplaatst. Hij komt voor in twee toestanden. In de input-mode geeft hij het goed zichtbare, kleine, knipperende rechthoekje op het scherm weer. In de programma-modus is de cursor niet zichtbaar en alleen virtueel aanwezig. Dat wil zeggen, dat hij alleen de actuele beeldschermpositie aangeeft. Wanneer er dus wordt gesproken over de cursorpositie, wordt altijd de positie van het eerstvolgende teken bedoeld.

SPC	Deze functie legt een tussenruimte vast tussen de huidige cursorpositie en de aansluitend te schrijven tekst. Ze dient dus voor de relatieve positionering van de beeldschermweergave.
TAB	TAB specificeert de kolom van de regel, waarvanaf de volgende output moet plaatsvinden. Hier wordt dus een absolute positionering ingesteld (vergelijk ook AT).
POS	Om de actuele cursorpositie binnen een regel te weten te komen, kan deze instructie worden gebruikt. Deze instructie geeft de mogelijkheid tot oriëntatie op het beeldscherm en is erg effectief in verband met de LIN-instructie van Simon's BASIC.
, / ;	Zoals u waarschijnlijk bekend is, wordt na ieder PRINT-statement een zogenaamde CARRIAGE-RETURN uitgevoerd. Dat wil zeggen, dat er een sprong plaatsvindt naar het begin van de volgende regel. Dit kan worden verhinderd door het toevoegen van een "," (komma) of een ";" (puntkomma). Terwijl de puntkomma een zuiver onderdrukkende functie heeft, om direkt zonder tussenruimte aan een PRINT-output een andere PRINT-output te kunnen voegen, wordt de volgende output na het toepassen van de komma automatisch op de volgende vaste tabpositie gezet.

We bespreken nu de extra instructies van Simon's BASIC.

9.1.1 CENTRE

Format	CENTRE str
Parameter	str: "tekenreeks" of stringvariabele met een lengte die kleiner is dan 40 posities
Voorbeeld	CENTRE A$ of CENTRE "Simon'S BASIC"
Functie	gecentreerde output van een string.

Met de instructie CENTRE heeft u de mogelijkheid de output op het beeldscherm gecentreerd vorm te geven. Onder gecentreerde output verstaan we het volgende. Veronderstel, dat u de titel 'Commodore 64' wilt weergeven, die in het midden van de eerste regel moet verschijnen. Normaal gesproken zou u eerst het aantal letters moeten tellen; dit van het aantal tekens per regel (40) aftrekken en het resultaat door 2 delen, om het aantal spaties ervoor te berekenen. Deze moeite kunt u zich besparen. De instructie CENTRE zet automatisch de spaties, die nodig zijn om de tekst in het midden te krijgen. De tekst mag niet langer zijn dan 39 tekens, anders verschijnt de weergave tamelijk willekeurig op het beeldscherm. De cursor staat daarna op de positie direkt achter het geschrevene, wat betekent, dat u eerst een Carriage-Return moet uitvoeren in de vorm van een PRINT of PRINT CHR$(13), wanneer u eerst in de volgende regel wilt schrijven. U kunt deze methode ook gebruiken wanneer u bijvoorbeeld met CENTRE " " alleen de cursor positioneert zonder iets te schrijven, om later vanuit deze positie tekens te plaatsen. Voegt u direkt achter een CENTRE-instructie een tweede toe, dan wordt het aantal spaties voor de lege plaatsen voor de eerste output weer opgenomen en in dezelfde afstand voor de tweede geplaatst. U hebt dus bij het optimale gebruik van dit waarschijnlijk onopzettelijke verschijnsel de mogelijkheid, verschillende delen van de output met constante afstand uit te voeren. Een voorbeeld:

```
100 REM ***********************
110 REM **                   **
120 REM ** CENTRE-VOORBEELD  **
130 REM **                   **
140 REM ***********************
150 REM
160 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
170 A$ = "SIMON'S BASIC"
180 B$ = "VOOR CBM 64"
190 CENTRE A$ : PRINT
200 CENTRE B$ : PRINT
210 CENTRE "*****************"
220 REM
230 REM     *****************
240 REM     **  AANVULLING  **
250 REM     *****************
260 REM
270 PRINT
280 FOR X = 17 TO 1 STEP -2
290 CENTRE DUP ("*",X) : PRINT
300 NEXT X
```

In het eerste deel van dit voorbeeld wordt alleen de titel gecentreerd afgebeeld. Het tweede deel, dat aan het programma wordt toegevoegd, geeft een aardig effect.

9.1.2 USE

Format	USE str1, str2
Parameter	str1: "tekenreeks" of stringvariabele van de vorm "###.###" of "## tekst. ### tekst"
	str2: "tekenreeks" of stringvariabele met numerieke waarde
Voorbeeld	USE "###.##"'129.345"
Functie	decimaalpunt-georiënteerde output van numerieke waarden.

Met de instructie USE wordt de mogelijkheid van een tabellarische output van decimale getallen gegeven. Dat betekent, dat u bijvoorbeeld afrekeningen mooi op decimaalpunt (-komma) onder elkaar op het beeldscherm kunt zetten, zonder eerst omslachtig van ieder getal te moeten vaststellen, waar de betreffende decimaalposities moeten worden ingericht. Het daarbij te gebruiken format wordt met str2 aangegeven. Ieder cijferkruis (#) stelt een decimaalplaats van het latere output-getal voor. Tot de punt in str2 wordt bij de output voor ieder cijferkruis een spatie vrijgehouden, voor het geval die niet door een cijfer wordt bezet. Hetzelfde geldt voor de plaatsen na de komma (punt). Is een decimaal getal langer dan het aangegeven getal van cijferkruisen, dan wordt dit op de laatste positie eenvoudig afgesneden. Let hier op en rond zonodig van te voren het getal af.

Natuurlijk kunt u op de boven onder parameter aangegeven plaatsen de betreffende tekst in de formataanwijzing invoegen. U moet dan wel rekening houden met de rol van de punt en de rol van het cijferkruis (#) voor het format. Deze formataanwijzing kan natuurlijk als eenvoudige "tekenreeks" of per van te voren geprepareerde stringvariabele in de instructie worden ingevoerd. Str 2 bevat het te formateren decimaalgetal, dat voorheen door de instructie STR$( ) in een string werd veranderd, of als "tekenreeks" werd aangegeven.

Er ontstaan problemen bij een gelijktijdig gebruik van tekst en formatering in een regel. In zo'n geval ontstaan op grond van een tekortkoming van het programma fouten in de formatering, wanneer de laatste decimaalplaatsen achter de komma (punt) gelijk nul worden. Wanneer hier dus onjuistheden optreden, is dat meestal niet uw schuld. In zo'n geval kunt u het beste de tekst eerst met PRINT schrijven en de formatering door middel van USE eraan toevoegen.

```
100 REM ********************
110 REM **                **
120 REM **  USE-VOORBEELD **
130 REM **                **
140 REM ********************
150 REM
160 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
170 A$ = "***.**"
180 A = 12.34
190 B = 230.12
200 USE "KANTOORBENODIGDHEDEN: " + A$,STR$(A) : PRINT " GLD"
210 USE "BOEKEN              : " + A$,STR$(B) : PRINT " GLD"
220 PRINT  "------------------------------"
230 USE "TOTAAL              : " + A$,STR$(A+B) : PRINT " GLD"
```

Dit voorbeeld beschrijft een interessante toepassing van de zojuist beschreven instructie. U kunt er vast zelf wel meer bedenken.

9.1.3 AT

Format	PRINT str1 AT(k,r) str2
Parameter	str1/2 twee uit te voeren "tekenreeksen" of stringvariabelen, waarbij str1 optioneel is, zodat deze string ook kan worden weggelaten
	k : kolom
	r : regel
Voorbeeld	PRINT AT(10,12) "CBM 64"

Functie		de gepositioneerde output van een string

Met AT in een PRINT-statement bestaat de mogelijkheid een onder str2 aangegeven string ("tekenreeks" of stringvariabele) willekeurig op het beeldscherm te plaatsen. Daarbij geven de parameters k en r steeds de kolom en de regel van het begin van de output aan. Dat wil dus zeggen, dat k en r de positie van de cursor bepalen voor de output (cursor-coördinaten). Daarbij geldt ook weer, dat andere weergaven op het beeldscherm direkt aan het laatste teken van de gepositioneerde output worden toegevoegd. Ook hier is met str2=" " (lege string) een gemakkelijke plaatsing van de cursor op het beeldscherm mogelijk. Indien gewenst kunt u met str1 natuurlijk ook de tekst nog vanaf de oude positie schrijven.

Een bijzonder interessante toepassing van de AT-instructie is het maken van low-resolution-graphics (LGR = grafiek met laagoplossend vermogen). Daarbij zijn als "punten" natuurlijk alle tekens van de Commodore 64 toegestaan (vergelijk de voorbeelden).

U merkt, dat de AT-instructie een uiterst nuttige en veelzijdige instructie is, waarvan de mogelijkheden pas bij het proberen langzamerhand duidelijk worden. Alleen door voortdurend zelf te proberen leert u de veelzijdigheid van deze instructies kennen.

Nog een opmerking met betrekking tot de syntax: het eerste haakje na AT hoort bij de instructie. Tussen AT en (mag dus geen spatie zitten!

```
100 REM ********************
110 REM **                **
120 REM ** AT-VOORBEELD-1 **
130 REM **                **
140 REM ********************
150 REM
160 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
170 PRINT AT(10,5) "DE 'AT'-"
180 PRINT AT(13,10) "FUNCTIE"
190 PRINT AT(16,15) "IN ACTIE"
```

```
100 REM ********************
110 REM **                **
120 REM ** AT-VOORBEELD-2 **
130 REM **                **
140 REM ********************
150 REM
160 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
170 FOR X=0 TO 24
```

```
180 PRINT AT(X,X) "SIMON'S BASIC";
190 NEXT X
200 PRINT AT(0,20) ""
```

```
100 REM ********************
110 REM **                **
120 REM ** AT-VOORBEELD-3 **
130 REM **                **
140 REM ********************
150 REM
160 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
170 FOR X=0 TO 39
180 PRINT AT(X , 12 * SIN(X/6) + 12) "**"
190 NEXT X
```

Deze voorbeelden verduidelijken enkele gebruiksmogelijkheden van deze instructie, van de eenvoudig gepositioneerde en eventueel geformateerde output van tekst tot en met het gebruik van grafieken. Omdat door de AT-instructie geen enkel teken op het beeldscherm werd gebracht, dat door de cursor op de gewenste plaats wordt gezet, bijvoorbeeld een spatie, heeft de instructie natuurlijk in samenhang met het gebruik van een printer alleen een functie, wanneer de instructie HRDCPY (beeldschermhardcopy; zie ook 11.2.2) wordt gebruikt. Bij direkte printuitvoer na invoer van CMD of PRINT # wordt hij genegeerd en str2 wordt zo aan de oude drukpositie toegevoegd.

9.1.4 LIN

Format	LIN (als functie)
Parameter	---
Voorbeeld	A=LIN
Functie	Het bepalen van de regel waar de cursor moet komen

LIN is een van die instructies, die niet op zichzelf staan, maar net als bijvoorbeeld SIN of SQR een functie weergeven, waarmee verder wordt

gewerkt, of die direkt in een geheugenplaats (bijvoorbeeld: A=LIN) kan worden opgeslagen.

LIN biedt u de mogelijkheid de beeldschermregel te definiëren waarin de cursor komt te staan. Het Commodore-BASIC biedt de mogelijkheid, de cursorpositie binnen een regel door middel van POS op te sporen. De betreffende regel blijft echter onbekend. Dit gebrek is hiermee verholpen. Met POS en LIN krijgt u de exacte cursorcoördinaat die voor berekeningen of weergave kan worden gebruikt. Let op de verschillende schrijfwijzen van deze beide instructies!

Voorbeeld:

```
100 REM *******************
110 REM **               **
120 REM ** LIN-VOORBEELD **
130 REM **               **
140 REM *******************
150 REM
160 PRINT CHR$(147); : REM BEELDSCHERM WISSEN
170 A = LIN
180 FOR X=1 TO 20
190 PRINT "DIT IS DE" LIN ". REGEL"
200 NEXT X
210 PRINT
220 PRINT "NU ZIJN DE REGELS" A "TOT" LIN "BESCHREVEN"
```

Dit voorbeeld is alleen om de werkwijze en het effect van de LIN-instructie te demonstreren. De gebruiksmogelijkheden worden vanzelf duidelijk tijdens het oefenen. Probeert u het maar eens!

9.1.5 PAUSE

Format PAUSE str, s of PAUSE s

Parameter str: karakterstring of stringvariabelen, die een commentaar bevatten

s : aantal wachtseconden

Voorbeeld PAUSE "VERTRAGING",3

Functie Vertraging van het programmaverloop gedurende s seconden.

Wat u anders moeizaam zou moeten bereiken door middel van wachtlussen, kunt u nu in een enkele instructie samenvatten: PAUSE. PAUSE dient ertoe, om het programmaverloop voor een bepaalde tijd die met s (= seconde) wordt aangegeven, te vertragen. Tegelijkertijd heeft u nog de mogelijkheid deze wachttijd van commentaar, hier met str aangegeven, te voorzien. De gebruiker van het programma kan de wachttijd met RETURN onderbreken om het programma voort te zetten. Mocht uw programma net bij deze instructie blijven steken, dan kan er geen onderbreking RUN/STOP worden verkregen. Gebruik daarvoor RUN/STOP met RESTORE of RETURN en dan RUN/STOP. Deze instructie kan natuurlijk erg effectief worden ingezet bij de foutcorrectie van programma's, wanneer bijvoorbeeld tijdens de PAUSE-functie tussenwaarden worden aangegeven of snelle, onoverzichtelijke gebeurtenissen worden vertraagd. Een voorbeeld:

```
100 REM *********************
110 REM **                 **
120 REM **  PAUSE-VOORBEELD **
130 REM **                 **
140 REM *********************
150 REM
160 PRINT CHR$(5): REM WITTE TEKST
170 FOR X=0 TO 15
180 POKE 53248 + 33 , X : REM ACHTERGRONDKLEUR
190 PRINT CHR$(147) "<RETURN> VOOR VERDER!"
200 PRINT
210 PAUSE "U ZIET KLEURNUMMER" + STR$(X) , 4
220 NEXT X
```

In dit voorbeeld wordt in een tijdsbestek van 4 seconden de achtergrondkleur 16 keer veranderd. U heeft nu de mogelijkheid de wachttijd te onderbreken door middel van RETURN.

9.1.6 Testopdrachten

1. Waar moet u op letten bij de instructie CENTRE?

- A) Achter CENTRE mag geen PRINT-statement staan.
- B) Twee CENTRE-instructies behoren door een PRINT te zijn gescheiden.
- C) De te centreren string mag maximaal een lengte hebben van 40 tekens.

2. Welke syntax van de verschillende instructies is juist?

- A) PRINT CENTRE "ABC"
- B) PRINT AT(2,4) "CBM 64"
- C) PRINT AT (2,4) "CBM 64"
- D) A=LIN*2
- E) PAUSE 5. "TYP RETURN"
- F) USE "### DM . ### PF",STR$ (123.34)

3. Welke instructies moet u invoeren om precies in het midden van het beeldscherm twee sterretjes te plaatsen?

- A) PRINT "**"
- B) CENTRE "**"
- C) PRINT AT(19,12) "**"
- D) A$= "**" : PRINT AT(0,12) "":CENTRE A$

9.2 Het knipperen van het beeldscherm

De ontwikkelaars van Simon's BASIC hebben ook gedacht aan aardige effecten: de knipperende weergave. Die kan worden gebruikt om bepaalde gedeeltes van de beeldschermweergave te laten opvallen, ook voor spelletjes. De toepassingen variëren van serieus programmeren tot leuke dingen voor de kinderen, die overigens ook heel zinvol kunnen zijn.

9.2.1 FLASH

Format	FLASH k,s
Parameter	k: kleur, die knippert met k van 0 tot 15
	s: snelheid, waarmee deze knippert, met s van 0 tot 255
Voorbeeld	FLASH 4,10
Functie	Het knipperen van een kleur op het beeldscherm

Met FLASH kunt u alle tekens, die in een bepaalde kleur k op het beeldscherm staan, continu met de snelheid s laten knipperen. Dat wil zeggen, dat alle tekens, die deze kleur hebben, voortdurend wisselen van normaal naar inverse. Inverse betekent, dat het teken er uitziet alsof het is gezet, nadat de toetsen CTRL met RVS ON tegelijkertijd zijn ingedrukt. De parameter k bepaalt de kleur van de tekens, die van de inverse in de normale toestand overgaan (en omgekeerd). Daarbij zijn voor k waarden tussen 0 (=zwart) en 15 (=grijs) toegestaan. De afzonderlijke waarden zijn als volgt aan de diverse kleuren toegekend:

0 = zwart	8 = oranje
1 = wit	9 = bruin
2 = rood	10 = lichtrood
3 = cyaan	11 = grijs 1
4 = paars	12 = grijs 2
5 = groen	13 = lichtgroen
6 = blauw	14 = lichtblauw
7 = geel	15 = grijs 3

De snelheid s wordt daarbij ongeveer (voor 1 seconde geldt s=58) in de eenheid van 1/60 seconde aangegeven (s=0 komt daarbij overeen met s=256). Er geldt overigens wel een kleine beperking. Wordt s kleiner dan 10, wordt er dus erg snel geknipperd, dan heeft de computer steeds veel te doen, want hij laat tenslotte het gehele beeldscherm knipperen. Daarom wordt het BASIC-programma geleidelijk trager. Gelijktijdig verandert echter ook de eenheid van s naar 1/30, 1/15 ... seconde. Het beeldscherm zal dus nooit iedere 1/60 seconde kunnen knipperen. Er doen zich eveneens problemen voor met de televisie of monitor, omdat de computer iedere 1/25 seconde een nieuw beeld maakt. De effecten, die ontstaan (strepen en onderbrekingen), kunt u wel eens bekijken.

Er is nog een belangrijk detail, waarop u moet letten. Is FLASH ingeschakeld en wordt er in de knipperkleur tekst toegevoegd op het beeldscherm, dan kan het voorkomen, dat delen van of de hele toevoeging precies "omgekeerd" knipperen, zich dus in de inverse toestand bevinden wanneer de overige tekst normaal is, en omgekeerd.

Dit komt doordat de computer in de normale modus schrijft (of inverse, wanneer de RVS ON in gebruik gesteld is), onafhankelijk van de actuele knipper-situatie van het beeld. Zodra de tekens op het beeldscherm staan, worden ze tegelijk met de andere tekens in de knipperkleur "omgedraaid". Om dezelfde reden knippert het inverse schrift na het inschakelen van FLASH precies in de tegenovergestelde fase.

9.2.2 OFF

Format	OFF
Parameter	geen
Voorbeeld	OFF
Functie	Het uitschakelen van de FLASH-modus.

Met OFF wordt het mogelijk, het knipperen dat door FLASH werd ingeschakeld, weer ongedaan te maken. Daarbij wordt steeds de huidige situatie van het knipperen in stand gehouden. Dat wil zeggen, dat de tekens, die in de knipperende kleur zijn gezet en op het moment dat OFF actief wordt inverse zijn weergegeven, na OFF inverse blijven en omgekeerd. Wanneer u precies wilt bepalen, in welke toestand uw tekstvenster is na het uitschakelen door middel van OFF, is het dus aan te bevelen het gehele venster met PRINT CHR$(147) te wissen en opnieuw weer te geven. Eigenlijk moet na iedere FLASH in het programma een OFF volgen, op z'n minst aan het einde van het programma, omdat het beeldscherm anders ook na beëindiging van het programma blijft knipperen.

Een voorbeeld van FLASH en OFF:

```
100 REM **********************
110 REM **                  **
120 REM **  FLASH-VOORBEELD **
130 REM **                  **
140 REM **********************
150 REM
160 POKE 53248 + 33 ,0 : POKE 53248 + 32 ,0 : REM ACHTERGROND- +
    VENSTERKLEUR
```

```
170 PRINT CHR$(147) CHR$(158) :REM BEELDSCHERM WISSEN +
    TEKSTKLEUR=GEEL
180 CENTRE "EVEN GEDULD, TOTDAT READY VERSCHIJNT":PRINT
190 CENTRE "OF <RETURN> INGEDRUKT HOUDEN"
200 PRINT AT(0,8) "" : REM CURSOR PLAATSEN"
210 CENTRE "**********************" : PRINT
220 CENTRE "**********************" : PRINT
230 CENTRE "***                ***" : PRINT
240 CENTRE "***                ***" : PRINT
250 CENTRE "***                ***" : PRINT
260 CENTRE "***                ***" : PRINT
270 CENTRE "**********************" : PRINT
280 CENTRE "**********************" : PRINT
290 PRINT AT(0,10) CHR$(31): REM BLAUW
300 CENTRE "                " : PRINT
310 CENTRE "   COMMODORE 64 " : PRINT
320 CENTRE "   SIMONS BASIC " : PRINT
330 CENTRE "                " : PRINT
340 FOR X=10 TO 1 STEP -1
350 FLASH 6,X : REM FLASH BLAUW
360 PAUSE 2
370 NEXT X
380 OFF
```

Dit voorbeeld geeft een bescheiden demonstratie van het gebruik van de instructies FLASH en OFF en diverse andere uitvoer-instructies. Nadat het beeldscherm en het venster zwart zijn gemaakt, verschijnt een geel cijferkruisraampje met een blauwe streep in het midden van het scherm. Vervolgens begint het blauwe schrift te knipperen. Dit geknipper wordt steeds sneller, totdat het bij de hoogste snelheid stopt.

9.2.3 BFLASH

Format	BFLASH s,k1,k2
Parameter	s : knippersnelheid
	k1/k2: twee kleuren, waartussen steeds wordt gewisseld

Voorbeeld BFLASH 30, 12, 14

Functie Het knipperen van het beeldschermraam tussen de twee aangegeven kleuren.

BFLASH maakt het mogelijk, diverse flits- en knippereffecten te verkrijgen. Zo'n flitseffect is spectaculair in spelletjes: een overwinning of een 'schot in de roos' kunnen ermee in het middelpunt van de aandacht worden gezet. Kortom, op een tamelijk simpele manier wordt een programma wat gevarieerder vorm gegeven. Met BFLASH wordt dit bereikt door het buitenste beeldschermvenster tussen twee kleuren heen en weer te laten knipperen. Net als bij FLASH kunt u de snelheid van het knipperen zelf bepalen. Daarvoor is de parameter s. De berekening is gelijk aan die bij FLASH. Om dus ongeveer 1 keer per seconde van kleur te verwisselen, moet u de waarde 60 voor s invoeren. Dat is ook hier slechts een richtwaarde. Omdat een kleurverandering van het beeldschermvenster erg snel plaatsvindt door een POKE, dat wil zeggen door het veranderen van een registerinhoud, is het mogelijk ook zonder veel tijdsverlies een snelle kleurverandering in te stellen. Dat werkt iets anders dan bij FLASH. Nu kunt u werkelijk iedere 1/60 seconde de kleur wisselen, maar omdat het beeld op het beeldscherm alleen 25 maal per seconde wordt vernieuwd, ontstaan bij erg snelle veranderingen bepaalde streep-effecten. Probeer dit maar eens!

De twee andere variabelen bepalen de kleuren waartussen u kunt wisselen, Deze kunnen waarden aannemen van 0 (=zwart) tot 15 (=grijs 3). Vergelijk de kleurentabel bij FLASH. Hierbij moet u op het volgende letten. Wordt er gewisseld tussen twee kleuren met een groot contrastverschil (extreem: zwart-wit), dan heeft dit ook effect op de rest van het televisiebeeld. De felle kleur ontneemt het binnenste venster de kleurintensiteit en/of de lichtsterkte, terwijl de donkere kleur ervoor zorgt, dat het binnenste deel er duidelijk uitspringt. Daarbij moet u bedacht zijn op een gelijktijdig, "mee-knipperen" van het tekstvenster, vooral bij een lichtarme achtergrondkleur (extreem: zwart). Dit effect is erg goed te zien bij voorbeeld 1 onder BFLASH 0.

9.2.4 BFLASH 0

Format BFLASH 0

Parameter ---

Voorbeeld BFLASH 0

Functie Het uitschakelen van de BFLASH-modus

BFLASH 0 heeft ongeveer dezelfde functie als OFF, alleen betrokken op BFLASH. Dat wil zeggen dat BFLASH 0 het knipperen van het beeldschermvenster uitschakelt. Daarbij blijft het venster relatief oncontroleerbaar in de laatst aangegeven kleur. Het is dus aan te bevelen, het kader in beginsel opnieuw te definiëren met POKE 53248 + 32,f (f betekent hier het gewenste kleurnummer), om er zo zeker van te zijn, welke vensterkleur u tenslotte overhoudt. BFLASH 0 moet in principe in uw programma worden gebruikt, op zijn laatst aan het einde, als u gebruik gemaakt heeft van BFLASH, omdat BFLASH anders ook na het beëindigen van het programma nog actief is.

Voorbeelden voor BFLASH en BFLASH 0:

Voorbeeld 1: Vervang in het FLASH-voorbeeld de regels 350 en 380 door de volgende:

```
350 BFLASH X,6,7 : REM WISSELING BLAUW-GEEL
380 BFLASH 0
```

Voorbeeld 2:

```
100 REM ****************************
110 REM **                        **
120 REM **  BFLASH - VOORBEELD 2  **
130 REM **                        **
140 REM ****************************
150 REM
160 POKE 53248 + 33,2 : REM ACHTERGRONDKLEUR ROOD
170 BFLASH 1,7,6 : REM FLITSKLEUR = GEEL - BLAUW
180 PRINT CHR$(5) CHR$(147) : REM TEKST WIT; BEELDSCHERM WISSEN
190 FOR X=0 TO 39
200 Y = X/2
210 PRINT AT( X , Y ) "*" : REM DIAGONAAL 1
220 PRINT AT( 39-X , Y ) "*" : REM DIAGONAAL 2
230 PRINT AT( X , 0 ) "*" : REM HORIZONTALE LIJN BOVEN"
240 PRINT AT( X , 20 ) "*" : REM HORIZONTALE LIJN ONDER"
250 PRINT AT( 0 , Y ) "*" : REM VERTICALE LIJN LINKS
260 PRINT AT( 39 , Y ) "*" : REM VERTICALE LIJN RECHTS
270 NEXT X : REM VOLGENDE PUNT
280 FLASH 1,15
```

```
290 PRINT AT(3,0) CHR$(150) : REM POSITIONEREN EN TEKST: LICHTROOD
300 CENTRE "BENT U VERBAASD?" : PRINT
310 CENTRE "KLEINIGHEID!"
320 PRINT AT(0,20) "" : REM READY. POSITIONEREN
330 PAUSE 6 : REM 6 SECONDEN WACHTEN
340 BFLASH 0
350 OFF : REM FLITSEN UIT
```

Dit is een tamelijk indrukwekkend voorbeeld van het mogelijke samenspel tussen FLASH, BFLASH, OFF, BFLASH 0 en enige andere instructies van Simon's BASIC. Zonodig kunt u eens proberen na te gaan, wat er in de verschillende programmadelen gebeurt, waarbij de regels 190-270 niet absoluut nodig zijn voor het begrip van de verschillende instructies. Enkele veranderingen kunnen gemakkelijk worden uitgevoerd, om met het programma te kunnen "spelen".

9.2.5 Testopdrachten

1. Welke instructie typt u in, wanneer u het venster wilt laten knipperen met de kleuren wit en blauw met een snelheid van ongeveer 1 wisseling per seconde?

   A) BFLASH 0
   B) FLASH 0,1
   C) BFLASH 60,1,0
   D) BFLASH 0,1,1

2. U wilt het knipperen van het roodgekleurde teken beëindigen, dat u van te voren had ingesteld. Welke instructies kiest u?

   A) BFLASH 0,0,0
   B) BFLASH 0
   C) OFF
   D) FLASH 0

3. Wat doet u, wanneer u slechts een deel van de tekst door middel van de FLASH-instructie wilt laten knipperen?

A) Ik schrijf eerst de tekst die moet knipperen, schakel het knipperen in, waarna ik de resterende, niet knipperende tekst invoer.
B) Ik handel precies tegengesteld aan wat onder a werd beschreven.
C) Ik schrijf de tekst die knippert in een andere kleur dan de niet-knipperende tekst en ik laat de kleur van de eerste tekst steeds oplichten.
D) Ik schrijf beide teksten in dezelfde kleur, laat echter door middel van FLASH 12,40 alleen de bovenste helft van het beeldscherm, waarop de eerste tekst staat, knipperen.

9.3 De achtergrond- en vensterkleur

Let nu op! We staan op het punt een belangrijke ontdekking te doen: Simon's BASIC heeft niet alleen de instructies van het originele Handboek, maar ook enige andere instructies, die helemaal niet zijn verwerkt in het Handboek. Het oorspronkelijk concept van Simon's BASIC kende nog zeven andere geplande commando's. Van deze zeven instructies functioneren er slechts drie. De andere vier zijn niet operationeel. Of de aangekondigde nieuwe versie hierin zal voorzien is onbekend. Het staat vast, dat u over drie andere instructies kunt beschikken. Deze hebben betrekking op de weergave op het beeldscherm. Voordat we ons daarmee bezighouden, moet eerst nog het volgende worden uitgelegd:

De Extended Colour-Modus:

Uw Commodore 64 heeft naast de bekende normale tekstmodus met een achtergrondkleur voor alle tekens en 15 kleuren, een andere tekstmodus, waarin voor elk teken een andere achtergrondkleur kan worden gekozen. Er staan u telkens 4 registers voor achtergrondkleuren, dus 4 vrij te kiezen achtergrondkleuren ter beschikking. Deze manier van weergave heet "de Extended Colour-Modus". De registers worden als volgt aangesproken:

achtergrondkleur 0 .... POKE 53248 + 33,ak0
achtergrondkleur 1 .... POKE 53248 + 34,ak1
achtergrondkleur 2 .... POKE 53248 + 35,ak2
achtergrondkleur 3 .... POKE 53248 + 36,ak3

Voor ieder teken staat 1 van de 4 achtergrondkleuren ter beschikking, die in het register kunnen worden veranderd met instructies, of direkt door inPOKEn van de betreffende waarden. Daaruit blijkt, dat het kleurregister 0 overeenkomt met het normale register voor het vastleggen van de achtergrondkleur. Parameter ak is de kleurcode van dat moment. In de bijlage kunt u de gegevens vinden. Hoe kan nu worden vastgelegd, welke van deze achtergrondkleuren de tekens van dat moment uiteindelijk hebben?

De bovenste 2 bits van iedere byte in het beeldschermgeheugen, dat eigenlijk alleen de beelschermcodes voor de afzonderlijke tekens in het tekstvenster opslaat (zie boven), leggen dit steeds voor ieder teken vast. Omdat echter deze beide bits normaal gesproken eveneens gebruikt worden om diverse tekens te coderen, heeft u in de Extended Colour-modus de beschikking over 64 tekens.

Alle grafische tekens en de modus van de klein- en grootgeschreven tekens, evenals de inverse tekens, worden hiervan het slachtoffer. Stuurt u echter toch hierop aan, dan zal een van de toegestane tekens met een andere achtergrondkleur verschijnen. Welke tekens dat zijn, kunt u lezen in de tabel van de beeldschermcodes (CBM 64-Gebruikershandboek blz.133-134). Daarvoor geldt de volgende toevoeging:

MSBs	Bildschirmcodes	Hintergrundfarbe
00	000 - 063	HF 0
01	064 - 127	HF 1
10	128 - 191	HF 2
11	192 - 255	HF 3

Onder MSBs (meest significante bits) worden de beide bovenste bits van het beeldschermgeheugen verstaan (bits 6 en 7). De beeldschermcodes zijn steeds decimale waarden. Vergelijk deze tabel met de aangegeven tabel met beeldschermcodes in het Handboek. Wilt u dus een B met de achtergrondkleur 7 = geel op het beeldscherm brengen, definieer dan het gewenste register voor achtergrondkleuren (hier 1) met de waarde 7 voor geel en POKE een 2 voor B vermeerderd met 64 voor het inschakelen van het register 1 op de overeenkomstige plaats in het beeldschermgeheugen. U kunt ook door middel van een PRINT-statement het teken de instructie CHR$(98) meegeven op het beeldscherm. In het programma zou dit er dan als volgt uitzien:

```
10 BCKGNDS 2,3,4,5 :REM EXTENDED COLOUR-MODUS AAN
20 POKE 53248+34,7 :REM ACHTERGRONDKLEUR 1=GEEL
30 POKE 1024,2+64 :REM TEKEN IN LINKERBOVENHOEK
```

of

```
10 BCKGNDS 2,3,4,5 :REM EXTENDED COLOUR-MODUS AAN
20 POKE 53248+34,7 :REM ACHTERGRONDKLEUR 1=GEEL
30 PRINT CHR$(98) :REM TEKEN OP DE CURSORPOSITIE
```

Na afloop van het programma bevindt u zich weer in de extended colour-modus, u kunt zo wat uitproberen. En dan komen we nu toe aan de instructies, waarop u waarschijnlijk al lang gewacht hebt.

9.3.1 COLOUR

Format	COLOUR vk, ak
Parameter	vk: vensterkleur (0-15)
	ak: achtergrondkleur (0-15)
Voorbeeld	COLOUR 6,7
Functie	Het bepalen van de kleur van het venster en van de achtergrond

De eerste van de drie nieuwe instructies biedt de mogelijkheid om op het beeldscherm de kleur van het venster of de kleur van de achtergrond te veranderen, zonder dat daarvoor omslachtige POKEs moeten worden ingevoerd. Het nadeel van POKEs is, dat de functie daarvan na verloop van enige tijd wordt vergeten als u geen REMs hebt toegepast.

Met vk, de eerste parameter die de functie van COLOUR bepaalt, kan de kleur van het buitenste venster op het beeldscherm worden bepaald. Voor vk moet de kleurcode worden ingevoerd, waarmee u de kleur voor het venster wilt definiëren. Deze waarde wordt dan automatisch in het betreffende register opgeslagen. Bij het vastleggen van het beeldschermvenster moet u erop letten, dat erg felle kleuren (net als de BFLASH-instructie) de lichtsterkte binnen het venster afzwakken, wat kan leiden tot onleesbaarheid van een tekst.

Een voorbeeld maakt dit duidelijk. Dit deel van de functie van de COLOUR-instructie komt dus overeen met de instructie:

POKE 53248 + 32,vk

De parameter ak bedient, zoals u waarschijnlijk al gedacht had, de achtergrondkleur van de tekst (achtergrondkleur 0). Daarmee wordt natuurlijk alleen de achtergrondkleur in de tekstmodus bedoeld. Voor de grafiek gelden andere instructies, terwijl de vensterkleur eveneens in de grafische modus kan worden veranderd. Wanneer de achtergrondkleur wordt bepaald, moet u erop letten, dat er kleuren gekozen worden, die een zo groot mogelijk contrast met de kleur van de tekst vormen, omdat er anders vervelende effecten ontstaan (bijvoorbeeld: rode interferentie op blauwe achtergrond). Dit deel van de instructie vervangt dus de instructie:

POKE 53248 + 33,ak

In z'n geheel zou men COLOUR kunnen vervangen door

POKE 53248 + 32,vk : POKE 53248 + 33,ak

Het voordeel van deze instructie is nu duidelijk. Een voorbeeld:

```
100 REM **********************
110 REM **                  **
120 REM **  COLOUR-VOORBEELD **
130 REM **                  **
140 REM **********************
150 REM
160 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
170 PRINT CHR$(31) : REM TEKSTKLEUR = BLAUW
180 PRINT AT(0,8) "" : REM CURSOR PLAATSEN
190 CENTRE"***********************":PRINT
200 CENTRE "**                   **":PRINT
210 CENTRE "**  WE VERANDEREN    **":PRINT
220 CENTRE "**  DE VENSTERKLEUR  **":PRINT
230 CENTRE "**                   **":PRINT
240 CENTRE "***********************":PRINT
250 REM
260 FOR VK=0 TO 15
270 COLOUR VK,0 : REM ACHTERGRONDKLEUR = ZWART/VENSTERKLEUR VERANDERD
280 PAUSE 1
290 NEXT VK
300 REM
310 PRINT AT(0,11) "" : REM CURSOR PLAATSEN
320 CENTRE "ACHTERGRONDKLEUR" : PRINT
330 REM
340 FOR AK=0 TO 15
```

```
350 COLOUR 0,AK : REM VENSTERKLEUR = ZWART/ACHTERGRONDKLEUR VERANDERD
360 PAUSE 1
370 NEXT AK
```

9.3.2 BCKGNDS

Format	BCKGNDS ak0,ak1,ak2,ak3
Parameter	ak0 : achtergrondkleurregister 0 definiëren (0-15)
	ak1 : achtergrondkleurregister 1 definiëren (0-15)
	ak2 : achtergrondkleurregister 2 definiëren (0-15)
	ak3 : achtergrondkleurregister 3 definiëren (0-15)
Voorbeeld	BCKGNDS 3,4,5,6
Functie	Het inschakelen van de Extended Colour-Modus en het bepalen van de 4 achtergrondkleuren

Voordat u met deze instructie aan het werk gaat, moet u eigenlijk eerst de inleiding van deze paragraaf (9.3) lezen, omdat u anders misschien de basis mist voor het begrip van de onderlinge samenhang. Deze instructie ontsluit de mogelijkheden van de Extended Colour-Mode. De CBM 64 heeft nog meer mogelijkheden, die tot op heden in slechts weinig programma's werden aangeboden. De betekenis van de Extended Colour-Mode wordt in de inleiding van deze paragraaf beschreven. Zo kunt u aan ieder beeldschermteken een eigen achtergrondkleur toewijzen. Daarvoor wordt voor ieder teken gekozen, uit welk register de achtergrondkleur moet komen. Tegelijkertijd is de hoeveelheid tekens beperkt tot 64. Met BCKGNDS bereikt u dit proces, dat alleen in de tekstmodus zichtbaar is. Dan worden de kleuren bepaald, die aan de 4 registers moeten worden toegekend. Deze vier kleuren worden met de parameters ak0, ak1, ak2 en ak3, die de kleurcodes van iedere kleur weergeven, gekozen. Is bijvoorbeeld ak0=7, dan krijgt het register 0 voor de achtergrondkleur de kleur geel als achtergrondkleur van alle tekens, die heel normaal werden ingevoerd. Enkele voorbeelden zullen dit verduidelijken:

Voorbeeld 1:

```
100 REM **************************
110 REM **                      **
120 REM **  BCKGNDS-VOORBEELD-1 **
130 REM **                      **
140 REM **************************
150 REM
160 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
170 PRINT CHR$(5) : REM TEKSTKLEUR = WIT
180 COLOUR 0,0 : REM VENSTERKLEUR = ZWART
190 BCKGNDS 2,3,6,7 : REM EXTENDED COLOUR MODUS INSCHAKELEN
200 REM EN ACHTERGRONDKLEUR BEPALEN
210 FOR X=0 TO 255
220 POKE 2*X + 1024,CODE : REM TEKENS PLAATSEN
230 POKE 2*X + 55296,0 : REM KLEUR VAN DE TEKENS IN HET VENSTER (ZWAF
240 CODE = CODE + 1 : REM VOLGENDE BEELDSCHERMCODE
250 NEXT X
260 PRINT AT(0,15) "" : REM CURSOR VOOR READY. POSITIONEREN
```

Dit voorbeeld geeft alle mogelijke tekens in alle 4 achtergrondkleuren, die voorheen door BCKGNDS werden gedefinieerd, op het beeldscherm weer. Daarbij zullen de regels 220 en 230 niet onbegrijpelijk zijn, wanneer u de uitleg van de instructie FCHR hebt gelezen. Eerst wordt hier de in regel 240 verhoogde code van het beeldschermteken op een plaats in het beeldschermgeheugen vastgelegd; vervolgens wordt de kleur bepaald in de desbetreffende geheugencel van het kleurvenster, om het teken daadwerkelijk zichtbaar te maken. Verander de afzonderlijke achtergrondkleuren maar eens door middel van BCKGNDS.

```
100 rem ************************
110 rem **                    **
120 rem **  bckgnds-voorbeeld **
130 rem **                    **
140 rem ************************
150 rem
160 print chr$(147) : rem beeldscherm wissen
170 print chr$(5) : rem tekstkleur = wit
180 colour 0,0 :rem venster- achtergrondkleur
= zwart
190 bckgnds 2,5,6,0 : rem extended
colour-modus inschakelen
200 rem en achtergrondkleur bepalen
210 rem kleine letters
220 centre "zoals u ziet, is het mogelijk":print
230 centre "om in de extended colour-modus":print
240 centre "volledig normaal te schrijven":print
250 pause 9
```

```
260 rem
270 rem de volgende tekens zijn alle ge<shift>
en zichtbaar te maken met <shift>< c>
280 print chr$(147) rem beeldscherm wissen
290 print:lege regel
300 centre "MAAR U BEZIT EEN GROOT AANTAL":print
310 centre "ACHTERGRONDKLEUREN DIE U":print
320 centre "VELE MOGELIJKHEDEN BIEDEN":print
330 pause 9
340 rem
350 print chr$(147) : rem beeldscherm wissen
360 print : lege regel
370 rv$=chr$(18) : rem reverse aan voor
achtergrondkleur
380 rem de volgende tekens zijn met inverse
ingevoerd
390 centre "u kunt voor elk teken een":print
400 centre "aparte achtergrondkleur":print
410 centre "van 4 mogelijkheden uitkiezen":print
420 pause 9
430 rem
440 print chr$(147) : rem beeldscherm wissen
450 print : rem lege regel
460 rem de volgende tekens zijn met <shift> en
inverse ingevoerd
470 ko$=chr$(172) : rem kommacode + 64 voor ak3
geeft het teken < c< >d
480 centre "MAAR U MOET ER WEL VOOR":print
490 centre "UITKIJKEN," + ko$ +"GEEN GRAFIEK-":print
500 centre "TEKENS IN UW TEKST TE GEBRUIKEN":print
510 pause 9
520 rem
530 print chr$(147) : rem beeldscherm wissen
540 print : rem lege regel
550 centre "natuurlijk KUNT "+ rv$ +"U ALLE" :print
560 print : lege regel
570 ro$=chr$(146): rem rv$ uit
580 h$= "aC" + rv$ +"hT" + ro$ +"eR" + rv$ +"gR"
+ ro$ +"oN"+ rv$ +"d "
590 h$= h$ + ro$ +"kL"+ rv$ +"eU"+ ro$ +"rE"+ rv$
+"n " :rem woord samenstellen
600 print at(10,4) h$
610 print : lege regel
620 centre "gelijktijdig"+ rv$ +"GEBRUIKEN"
630 pause 15
640 rem
650 print chr$(147) : rem beeldscherm wissen
660 print :lege regel
670 centre"kijkt u nu eens naar de ":print
```

```
680 centre"programmalisting in deze modus;":print
690 centre"kunt u het verklaren?":print
700 pause 9
710 rem
720 list
```

Zoals u vast wel gemerkt zult hebben, werd deze listing in kleine letters afgedrukt. Dit heeft een speciale reden. De alternatieve karakterset van de Commodore 64 werd gebruikt, wat wil zeggen dat alles, wat anders groot geschreven is, nu in kleine letters wordt weergegeven en alle grafische tekens als hoofdletters verschijnen. Omdat we echter deze geSHIFTe tekens nodig hebben om alle achtergrondkleuren aan te spreken, hebben we voor deze output gekozen. Schakel dus het apparaat voor de input van het programma om op de groot/klein-modus, door het gelijktijdig indrukken van de Commodore- en de SHIFT-toets. (De Commodore-toets wordt in de REM-regels van het programma met C= weergegeven).

In de Extended Colour-Mode ziet het er heel anders uit. De verklaring luidt als volgt. Alle normaal geschreven tekens worden ook normaal met de gebruikelijke achtergrondkleur 0 aangegeven; dat zijn in het programma de niet geSHIFTe kleine letters: CHR$(32) tot CHR$(95). Alle tekens, die met SHIFT of met de Commodore-toets worden ingevoerd: CHR$(96) tot CHR$(191), zonder CHR$(128) tot CHR$(159), worden daarentegen voorzien van de achtergrondkleur 1.

Welke toetsen komen met bepaalde tekens overeen? Dat ziet u hier.

- 1. Zoek de code van het gewenste teken op in de tabel met beeldschermcodes. Het teken mag geen code bevatten, die groter is dan 63, anders kan het niet weergegeven worden.

- 2. Tel 64 bij de verkregen code op en controleer in dezelfde tabel, welk teken met deze nieuwe code overeenkomt.

- 3. Zoek nu dat teken in de ASCII-tabel van het Handboek. Dan heeft u de ASCII-waarde die moet worden aangegeven om het gewenste teken in de achtergrondkleur 1 (of 3) te krijgen.

Alle letters kunnen zonder deze gecompliceerde handeling gemakkelijker in de gewenste achtergrondkleur worden gezet door het gelijktijdig indrukken van de SHIFT-toets. Bij de cijfers of leestekens moet u echter wel volgens de bovengenoemde manier handelen (vergelijk met regel 470 van het tweede voorbeeld). Wilt u nu een teken met achtergrondkleur 2 weergeven, dan moet u net zo handelen als wanneer u met de kleur 0 zou willen schrijven, alleen moet u hier alle tekens in de inverse-modus (na input van RVS ON) zetten. Achtergrondkleur

3 wordt door de inverse output van de tekens weergegeven, die anders voor kleur 1 werden gebruikt.

Voorbeeld: Het cijfer 9 en de letter C weergegeven in de 4 achtergrondkleuren.

```
Achtergrondkleur 0 : PRINT "9"
                     PRINT "c"
Achtergrondkleur 1 : PRINT CHR$(185)
                     PRINT "C"
Achtergrondkleur 2 : PRINT CHR$(18) "9"
                     PRINT CHR$(18) "c"
achtergrondkleur 3 : PRINT CHR$(18) CHR$(185)
                     PRINT CHR$(18) "C"
```

Mocht u na deze beschrijving er toch nog moeite mee hebben dit te begrijpen, gooi het boek dan niet gefrustreerd in de prullemand, maar sla dit hoofdstuk eerst even over. U kunt het na wat meer oefening altijd nog een keer proberen. De voorbeelden moet u wel goed doornemen, anders lukt het niet.

9.3.3 NRM

Format	NRM
Parameter	-----
Voorbeeld	NRM
Functie	Het uitschakelen van de Extended Colour-Modus

Nadat u alles hebt doorgewerkt, wat u over de Extended Colour-Modus zou moeten weten, ontbreekt er nog slechts een deel van de puzzel: het uitschakelen ervan. Deze functie neemt de instructie NRM (voor normaal) over, die tegelijkertijd de laatste functionele, tot nu toe onbekende instructie beschrijft. Met NRM wordt echter alleen de genoemde modus uitgeschakeld. De achtergrondkleur 0, die nu weer de enige achtergrondkleur is, blijft daarbij behouden. Ze kan nu weer door middel van COLOUR worden veranderd. NRM zullen we nog een paar keer tegenkomen, omdat deze enkele opties heeft:

1. Het uitschakelen van de Extended Colour-Modus
2. Het uitschakelen van de grafische modus
3. Het uitschakelen van de MEM-modus
   (dat wil zeggen het terugschakelen naar de oude karakterset)
4. Het inschakelen van de hoofdletter/grafiek modus.

Voorbeeld: Voer na afloop van het tweede voorbeeld onder 9.3.2 NRM in.

9.3.4 Testopdrachten

1. Stel, u wilt de achtergrondkleur wit en de kleur van het venster geel hebben. Welke instructies moeten dan worden ingevoerd?

   A) COLOUR 1,7
   B) COLOUR 7,1
   C) BCKGNDS 1,7
   D) POKE 53248 + 33,1 : POKE 53248 + 32,7

2. Wanneer u het teken "A" met de achtergrondkleur 2 wilt weergeven, hoe moet de output dan worden gericht?

   A) PRINT CHR$(65)
   B) PRINT CHR$(97)
   C) PRINT CHR$(18) CHR$(65)
   D) PRINT CHR$(18) CHR$(97)

9.4 Sturing: het beeldscherm.

In deze paragraaf wordt een reeks goed bruikbare instructies besproken die betrekking hebben op het beeldscherm. U heeft daarmee de

beschikking over snelle en effectieve mogelijkheden om de inhoud van het beeldscherm te beïnvloeden. Maar er treden problemen op wanneer u niet precies op de hoogte bent van de opbouw van het beeldscherm. Daarom volgt nu eerst een korte inleiding over de structuur en werking van het tekstvenster.

De computer slaat alle tekens op in het beeldschermgeheugen, zodat hij alle uitvoer die op het beeldscherm moet staan, kan onthouden. Hij moet dit beeld tenslotte op uw televisie iedere seconde 25 keer opstellen, zodat u het voortdurend kunt bekijken. Dit beeldschermgeheugen omvat ongeveer 1 Kbyte (=1024 tekens) en ligt in het geheugen vanaf $400 tot $7E7 (=1024 tot 1023 decimaal). Alle beeldschermtekens zijn apart gecodeerd. Dit is u waarschijnlijk al bekend van de ASCII-codering. De beeldschermcodes komen echter niet overeen met de gebruikelijke ASCII-codering, maar hebben voor ieder teken een eigen code. Daardoor kan onderscheid worden gemaakt tussen inverse en normale tekens, want hoe kan de computer anders uit een code opmaken, of hij het teken nu inverse of normaal op de monitor moet brengen? Een tabel met beeldschermcodes staat in het User Manual op de pagina's 132, 133 en 134. Wanneer bij de normale beeldschermcodes steeds 128 wordt opgeteld, krijgt u hetzelfde teken in de inverse vorm.

Voorbeeld:

```
10 PRINT CHR$(147) CHR$(158):REM BEELDSCHERM WISSEN - KLEUR = GEEL
20 FOR X= 1 TO 100
30 PRINT " "; :REM KLEUR OP BETREFFENDE PLAATS ZETTEN
40 NEXT X
50 FOR X=1024 TO 1124
60 POKE X,1:REM CODE VOOR A I/H BEELDSCHERMGEHEUGEN POKEN
70 NEXT X
```

In dit programma wordt linksboven in de hoek honderd keer een "A" in het beeldschermgeheugen geschreven. In de regels 10 tot 40 wordt eerst door middel van spaties de kleur bepaald (wat later nog wordt uitgelegd) en vervolgens wordt in de eerste honderd geheugencellen van het beeldschermgeheugen de waarde 1 opgeslagen als code voor het teken "A". Naast het teken zelf moet echter voor ieder teken ook de bijbehorende kleur in het geheugen worden opgeslagen. We moeten immers tekst in verschillende kleuren kunnen drukken. Hiervoor heeft de Commodore 64 een speciaal kleurgeheugen of kleur-RAM met dezelfde grootte als het beeldschermgeheugen, dat de nodige informatie bevat. Dit kleur-RAM ligt van $D800 tot $DFFF en kan net als het beeldschermgeheugen door POKE en PEEK bewerkt worden. Iedere byte in het kleur-RAM bepaalt de kleur van een bijbehorende teken in het beeldschermgeheugen. De achtergrondkleur van het tekstbeeld wordt door een enkel register van de zogenaamde videochip bepaald.

De videochip of VIC is het IC in uw computer, dat het beeld op de monitor weergeeft. Het register ligt in het geheugen op de plaats 53248+33 en kan bijvoorbeeld worden veranderd door

... POKE 53248+33,0 : REM ACHTERGRONDKLEUR = ZWART

Hierover meer in het hoofdstuk over graphics.

Een belangrijk probleem bij enkele instructies, die in deze paragraaf worden genoemd, is het volgende. Wanneer tijdens het programma het beeldscherm wordt gewist, worden ook het beeldschermgeheugen en het kleur-RAM gewist. Op iedere plaats in het kleur-RAM komt dan de code voor de achtergrondkleur te staan. Daarom ziet u dan niets, wanneer u weer iets in het beeldschermgeheugen wilt noteren, omdat het teken, dat eigenlijk op het beeldscherm zou moeten verschijnen dezelfde kleur heeft als de achtergrond. Wanneer u nu met de cursor over de plaats heengaat, waar u het teken verwacht te vinden, of als u de achtergrondkleur verandert, komt het weer tevoorschijn. Pas, als de betreffende plaats in het geheugen met een contrastkleur is gevuld, wordt het teken normaal zichtbaar.

Voorbeeld:

```
10 PRINT CHR$(147):REM BEELDSCHERM WISSEN
20 POKE 1024,2:REM EEN B I/H BEELDSCHERMGEHEUGEN
30 STOP :REM KAN OOK WEGGELATEN WORDEN
40 POKE 55296,0:REM EEN ZWART' TEKEN
```

Beweeg de cursor na afloop naar de linker bovenhoek. Daar bevindt zich een B. Wis nu regel 30 en start het programma opnieuw. Nu wordt de kleur-byte in het kleur-RAM geplaatst en verschijnt de B in het zwart. Tot zover bezit u voldoende informatie om met de volgende instructies aan het werk te kunnen gaan.

9.4.1 FCHR

Format FCHR r,k,b,h,c

Parameter: r : regel (0-24) en

k : kolom (0-39) van de linker bovenhoek van het

veld

b : breedte en

h : hoogte van het veld

c : definieert de te plaatsen karakters (0-255)

Voorbeeld — FCHR 3,5,7,9,1

Functie — Het vullen van een gebied op het beeldscherm met een teken

Door middel van FCHR (van FILL CHARACTERS) wordt het mogelijk een heel beeldschermgebied te vullen met een teken waarvan de beeldschermcode met c wordt aangegeven. U moet altijd een teken en een rechthoek definiëren die door de afzonderlijke parameters worden bepaald. Daarbij geven de beide eerste parameters de absolute positie van de linker bovenhoek op het beeldscherm aan. Parameter r geeft de regel aan, k de kolom van de linker bovenhoek. Dit zijn de begincoördinaten van de rechthoek. Met h en b wordt de hoogte en breedte van een veld bepaald, waarbij steeds in eenheden van een teken wordt gemeten. Dat wil zeggen: h of b geven de relatieve grootte van de rechthoek van tekens aan. Relatief met betrekking tot het door r en k bepaalde "beginpunt". De parameters h en b mogen niet groter worden dan het aantal regels of kolommen van het beginpunt tot de rand van het beeldscherm. Begint u dus met de rechthoek in regel 10/ kolom 10, dan mag h maximaal de waarde 14 aannemen, omdat er nog maar 14 regels tot aan de benedenrand van het beeldscherm vrij zijn. Verder mag b hoogstens 29 worden, omdat dit het aantal vrije tekens is tot aan de rechterkant. Wanneer deze waarden worden overschreden, of voor een van de parameters de waarde 0 wordt ingevoerd, wordt de foutmelding BAD MODE gegeven. Zoals u ziet, vervangt de instructie FCHR de volgende BASIC-routine:

```
170 FOR Y=R TO H:REM REGELLUS
180 FOR X=K TO B:REM KOLOMLUS
190 POKE 1024 + 40*Y + X,C: REM TEKEN PLAATSEN
210 NEXT X,Y :REM VOLGENDE KOLOM/REGEL
```

Hier betekenen r, h, k, b, c nog steeds respectievelijk regel, hoogte, kolom, breedte, code). Door regel 190 wordt het adres uit de actuele (dat wil zeggen, op dat ogenblik behandelde) regel (Y) en kolom (X) berekend: 1024 is het beginadres van het beeldgeheugen. Daarbij worden voor iedere regel 40 tekens opgeteld (iedere regel biedt immers plaats voor 40 tekens). Tot slot wordt het aantal tekens van de aangebroken regel opgeteld en het adres van het beeldschermteken is

berekend. Deze formule (adres=1024 + 40*regel + kolom) kan bij de berekening van het adres steeds worden gebruikt, wanneer direkt in het beeldschermgeheugen wordt gePOKEt. Wanneer u nu eens een snelheidsvergelijking wilt maken, volgt hier een voorbeeld van een FCHR-vervanging:

```
100 rem ********************
110 rem **                **
120 rem ** fchr-vervanging **
130 rem **                **
140 rem ********************
150 rem
160 r=1 : h=23 : k=2 : b=35 : c=1 : rem parameter
170 for y=r to h : rem regellus
180 for x=k to b : rem kolomlus
190 poke 1024 + 40*y + x,c : rem teken plaatsen
200 rem poke 55296 + 40*y + x,15: rem kleur instellen
210 nextx,y : rem volgende kolom/regel
```

Wanneer u dit kleine programma hebt uitgeprobeerd, zult u gemerkt hebben, dat er bij het wissen van het beeldscherm weinig gebeurt. Dit heeft te maken met de beschreven scheiding van beeldscherm- en kleurgeheugen. Om die reden werd in regel 200 een toevoeging ingevoerd (die op dat tijdstip nog niet actief was, omdat deze in een REM-regel stond): de kleurbepaling. Haal alleen de eerste REM uit regel 200 weg en laat het programma weer lopen, dan zult u meer succes hebben. Deze instructie is eigenlijk een klein voorproefje op de instructie FCOL die in de volgende paragraaf wordt besproken. We gebruiken een combinatie van de bij elkaar behorende instructies FCHR en FCOL. Daarbij wordt een formule gebruikt die kan worden toegepast bij de berekening van de kleur-RAM-positie die bij een teken hoort. Deze formule lijkt op die van het beeldgeheugen. Ze verschillen alleen in hun basisadres. Terwijl het beeldgeheugen begint bij het adres 1024, begint het kleur-RAM bij 55296. In dit voorbeeld werden de parameters, die bij de instructie FCHR horen, in regel 160 gezet. Verandert u dit eens en probeer deze instructie zelf. De beeldschermcode van een teken kunt u opzoeken in het CBM 64-Gebruikershandboek (blz.133-134).

Voorbeeld:

```
100 rem ********************
110 rem **                **
120 rem **  fchr-voorbeeld **
130 rem **                **
140 rem ********************
```

```
150 rem
160 poke 53248 + 33,1 : rem achtergrondkleur = wit
170 print chr$(147) : rem beeldscherm wissen
180 centre  "op het moment is het beeldscherm leeg " : print
190 pause 3
200 fchr 3,0,40,22,1 : rem teken (a) zonder kleur plaatsen
205 fcol 3,0,40,22;1
210 rem vanaf regel 3, kolom 0 met breedte 40 en hoogte 22
220 print at(0,5) ""; : rem cursor positioneren
230 centre "nu worden de regels 3 t/m 24 met tekens"
240 centre "zonder kleur gevuld. d.w.z. u ziet"
260 centre "niets!"
270 pause 6
280 print at(0,10) ""; : rem cursor positioneren
290 centre "nu gaan we de achtergrondkleur       "
300 centre "veranderen en alles wordt duidelijk"
330 pauseR 6
340 poke 53248 + 33,14 : rem achtergrondkleur = lichtblauw
350 pause 6
360 poke 53248 + 33,1 : rem achtergrondkleur weer wit
370 print at(0,15) ""; : rem cursor positioneren
380 centre "  wanneer u met de cursor over      "
390 centre "  het beeldscherm gaat, ziet u      "
400 rem lege regel
410 centre "    wat er aan de hand is!    "
```

We denken dat dit voorbeeld voldoende voor zichzelf spreekt.

9.4.2 FCOL

Format	FCOL r,k,b,h,f
Parameter	r : regel (0-24) en
	k : kolom (0-39) van de linker bovenhoek van het veld
	b : breedte van het veld
	h : hoogte van het veld

f : definieert de kleur van het in het veld af te beelden teken

Voorbeeld FCOL 5,4,5,3,8

Functie Het bepalen van de tekstkleur in een gedeelte van het beeldscherm

Voordat u deze paragraaf doorneemt, is het beslist noodzakelijk eerst de inleiding van deze paragraaf 9.4 en de beschrijving van de FCHR-instructie te lezen, omdat beide instructies FCHR en FCOL nauw met elkaar samenhangen. Met de eerste 4 parameters van de FCOL-instructie, die dezelfde functie hebben als bij de FCHR-instructie, wordt de positie en grootte van de rechthoek die met een bepaalde tekstkleur moet worden gevuld, aangegeven. Dat wil zeggen dat het kleur-RAM, dat de kleur van ieder afzonderlijk teken op het beeldscherm definieert, op deze plaats met de door f aangegeven kleur (0-15) wordt gevuld. Ieder teken in dit gebied krijgt dus nu de gewenste kleur f. Wilt u echter nog een nieuwe tekst erbij schrijven, dan wordt door middel van de PRINT-instructie de kleur van het teken in het kleur-RAM voor de betreffende tekens opnieuw ingevoerd, de nieuwe tekst krijgt dus ook een nieuwe kleur. Moet deze nieuwe tekst de oude kleur uit het kleur-RAM overnemen, dan moet u direkt het beeldschermgeheugen op de hierboven beschreven manier met POKE (of met FCHR) aanspreken. De gevolgen van de FCOL-instructie blijven beperkt tot tekst in het aangesproken gebied van het beeldscherm. Natuurlijk kan de FCOL-instructie net als FCHR worden vervangen door een kleine BASIC-routine, die gemakkelijker te begrijpen is. De toelichting daarop komt overeen met de toelichting die bij FCHR werd gegeven.

```
170 FOR Y=R TO H              : REM REGELLUS
180 FOR X=K TO B              : REM KOLOMLUS
190 POKE 55296 + 40*Y + X,F   : REM KLEUR INSTELLEN
200 NEXT X,Y                  : REM VOLGENDE REGEL/KOLOM
```

Let er ook hier op, dat u weinig van deze instructie ziet, wanneer u geen tekst op het beeldscherm hebt. Als voorbeeld van het effect van de FCOL-instructie kan het volgende programma dienen.

```
100 REM ***********************
110 REM **                   **
120 REM ** FCOL-VOORBEELD-1  **
130 REM **                   **
140 REM ***********************
150 REM
160 POKE 53248 + 33,4 : REM ACHTERGRONDKLEUR = CYAAAN
```

```
170 PRINT CHR$(147); : REM BEELDSCHERM WISSEN
180 PRINT CHR$(5) : REM TEKENKLEUR = WIT
190 CENTRE "OP HET OGENBLIK IS HET BEELDSCHERM CYAAN" : PRINT
200 PAUSE 3
210 PRINT AT(0,0) ""; : REM CURSOR BOVEN POSITIONEREN
220 CENTRE "NU WORDEN DE REGELS 01 TOT 24 MET" : PRINT
230 CENTRE "DE KLEUR BLAUW GEVULD. D.W.Z. U" : PRINT
240 CENTRE "ZIET ALLEEN DE BESTAANDE" : PRINT
250 CENTRE "TEKENS VERANDERD." : PRINT : PRINT
260 REM LEGE REGEL
270 PAUSE 8
280 PRINT CHR$(159); : REM TEKENKLEUR = CYAAN
290 CENTRE "OPGELET!!!" : PRINT : PRINT
300 REM LEGE REGEL
310 PAUSE 1
320 FCOL 1,0,40,24,6 : REM KLEUR (BLAUW) ZONDER TEKEN PLAATSEN:
330 REM VANAF REGEL 0, KOLOM 0, MET EEN BREEDTE VAN 40 EN EEN HOOGTE
    VAN 24
340 PRINT CHR$(5); : REM TEKENKLEUR = WIT
350 CENTRE "ALLES, WAT NU GESCHREVEN WORDT," : PRINT
360 CENTRE "WORDT WEER IN DE OORSPRONGSKLEUR" : PRINT
370 CENTRE "GEPLAATST."
```

Vervolgens wordt het gebruik van beide instructies (FCOL, FCHR) samen gedemonstreerd:

```
100 REM *************************
110 REM **                     **
120 REM **  FCHR/FCOL-VOORBEELD **
130 REM **                     **
140 REM *************************
150 REM
160 POKE 53248 + 33,0 : REM ACHTERGRONDKLEUR = ZWART
170 PRINT CHR$(147) : REM BEELDSCHERM WISSEN.
180 FOR X=0 TO 35 STEP 5
190 Y = Y+2 : REM REGELTELLER
200 FCHR Y,X,5,5,Y
210 REM BLOK VAN TEKENS MET DE BEELDSCHERMCODE Y PLAATSEN
220 FCOL Y,X,5,5,Y/2
230 REM BLOKKLEUR MET KLEUR Y/2 INSTELLEN
240 NEXT X
250 PAUSE 6
260 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
270 Y=0 : REM BEGINKLEUR = ZWART
280 FCHR 0,0,40,25,128+32
290 FCOL 0,0,40,25,6
```

```
300 REM BEELDSCHERM MET INVERSE BLAUWE SPATIES VULLEN
310 FOR X=4 TO 34 STEP 2
320 FCOL 5,X,2,15,Y : REM EEN KOLOM KLEUREN
330 Y = Y+1 :REM VOLGENDE KLEUR
340 NEXT X
```

9.4.3 FILL

Format	FILL r,k,b,h,c,f
Parameter	r : bepaalt de regel (0-24)
	k : de kolom (0-39) van de linker bovenhoek van het veld
	b : de breedte van het veld
	h : de hoogte van het veld
	c : definieert de te plaatsen karakters (0-255)
	f : de in te voeren tekstkleur (0-15)
Voorbeeld	FILL 3,5,7,9,2,11
Functie	Het bepalen van een tekstkleur in een gedeelte van het beeldscherm

Voor het bestuderen van deze instructie moet eigenlijk eerst heel paragraaf 9.4 worden gelezen. Zoals u heeft kunnen zien, hebben we steeds twee instructies nodig, wanneer we op een bepaalde positie een teken in een bepaalde kleur willen plaatsen. Maar de FILL-instructie maakt het mogelijk een blok met gekleurde tekens te vullen, omdat tegelijkertijd de kleur ook wordt ingevoerd. De FILL-instructie vervangt de FCHR- en FCOL-instructie samen met dezelfde parameters. Daarbij kunt u de betekenis van de afzonderlijke parameters onder FCHR en FCOL nalezen.
In plaats van

```
100 FCHR R,K,B,H,C ; FCOL R,K,B,H,F
```

kunt u nu vereenvoudigen tot:

```
100 FILL R,K,B,H,C,F
```

Deze beide regels vervullen precies dezelfde functie. Ook gelden dezelfde aanwijzingen die reeds werden geplaatst bij beide bekende instructies. Voorbeelden:

Vervang in het tweede voorbeeld onder FCOL (FCHR/FCOL-voorbeeld) de regels 280 en 290 door:

```
285 FILL 0,0,40,25,128+32,6
```

en de regels 200 en 220 door:

```
200 FILL Y,X,5,5,Y,Y/2
```

```
100 REM ***********************
110 REM **                   **
120 REM **  FILL-VOORBEELD 2 **
130 REM **                   **
140 REM ***********************
150 REM
160 REM     ******************
170 REM     **              **
180 REM     **  GRIDRUNNER  **
190 REM     **              **
200 REM     ******************
210 REM
220 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
230 POKE 53248 + 33,6 : REM ACHTERGRONDKLEUR = BLAUW
240 POKE 53248 + 32,0 : REM VENSTERKLEUR = ZWART
250 FOR X=1 TO 100
260 FILL 0,10,20,25,119,7 : REM CHR$(183)
270 FILL 0,10,20,25, 69,7 : REM CHR$(100)
280 FILL 0,10,20,25, 64,7 : REM CHR$( 96)
290 FILL 0,10,20,25, 70,7 : REM CHR$(102)
300 FILL 0,10,20,25,111,7 : REM CHR$(175)
310 NEXT X
```

Is dat geen indrukwekkend voorbeeld?

9.4.4 MOVE

Format	MOVE r,k,b,h,rd,kd
Parameter	r : bepaalt de regel (0-24),
	k : de kolom (0-39) bepaalt de linker bovenhoek
	h : geeft de hoogte aan
	b : de breedte
	rd: noemt de regel (0-24) van de nieuwe positie
	kd: de kolom van de nieuwe positie
Voorbeeld	MOVE 4,5,2,3,10,11
Functie	Het kopiëren van een deel van het beeldscherm

Met de MOVE-instructie bent u in staat delen van het beeldscherm op een andere plaats te dupliceren. U kunt dus een deel van het beeldscherm prepareren en vervolgens zonder moeite overbrengen op een ander deel van het tekstvenster. De parameters r, k, b en h zijn reeds bekend van de FCHR-/ FCOL- en FILL-instructies. Deze bepalen het te kopiëren gebied. Dit gebied is dus weer een rechthoek. Met rd en kd kiest u regel en kolom van de linker bovenhoek van het gebied, waar het duplicaat moet worden ingevoegd (dus net als de parameters r en k). Wanneer dat echter niet meer op het beeldscherm past, dus wanneer het gekopieerde buiten het scherm valt, wordt de foutmelding BAD MODE geproduceerd. Hier moet jammer genoeg een beperking worden vermeld. Versie V2 van Simon's BASIC vertoont hier een vervelende tekortkoming. De MOVE-instructie kan weliswaar worden gebruikt zonder een foutmelding, maar wanneer voor rd of kd variabelen of wiskundige uitdrukkingen worden toegepast, blijkt dat dit zinloze resultaten geeft. Er wordt slechts gedeeltelijk of zelfs helemaal niet gekopieerd. Daarom kunt u deze parameters alleen in de vorm van cijfers aangeven. Deze fout in Simon's BASIC is natuurlijk irritant, maar in zo'n ingewikkeld programma kunnen fouten bijna niet worden vermeden. Desondanks is deze instructie toch een erg handig hulpmiddel bij het genereren van tekstvensters.

Om de werking van de MOVE-instructie te illustreren, is dit kleine voorbeeld op z'n plaats:

```
100 REM **********************
110 REM **                  **
120 REM **   MOVE-VOORBEELD **
130 REM **                  **
140 REM **********************
```

```
150 REM
160 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
170 PRINT "DIT GEDEELTE "
180 PRINT "VAN HET BEELD-"
190 PRINT "SCHERM WORDT "
200 PRINT "GEDUPLICEERD "
210 PAUSE 4
220 MOVE 1,0,11,4,1,28
230 PAUSE 1
240 MOVE 1,0,11,4,5,14
250 PAUSE 1
260 MOVE 1,0,11,4,9,0
270 PAUSE 1
280 MOVE 1,0,11,4,9,28
290 PAUSE 1
300 MOVE 1,0,11,4,13,14
310 PAUSE 1
320 MOVE 1,0,11,4,17,0
330 PAUSE 1
340 MOVE 1,0,11,4,17,28
350 PAUSE 1
```

9.4.5 INV

Format	INV r, k, b, h Parameter r : regel (0-24) en k : kolom (0-39) van de linker bovenhoek van het veld b : breedte van het veld h : hoogte van het veld
Voorbeeld	INV 5, 6, 3, 4
Functie	Inverteren van een deel van het tekstvenster

INV biedt de mogelijkheid om een heel gebied van het beeldscherm te inverteren. De syntax van de parameters, die op het INV-commando volgen, zal inmiddels wel bekend zijn van alle andere blok-instructies (FCHR, FCOL, FILL, en MOVE). Zij definiëren het deel van het

beeldscherm dat wordt behandeld. De functie van de INV-instructie is vrij eenvoudig te verklaren. Ze lijkt op de FLASH-instructie, die alle beeldschermtekens van een bepaalde kleur met een bepaalde snelheid laat knipperen van de inverse situatie naar de normale, en omgekeerd. Met INV zullen alle tekens, die van te voren normaal (dus met RVS OFF) op het beeldscherm zijn geschreven worden geïnverteerd, dus net alsof deze met RVS ON zouden zijn ingevoerd. Omgekeerd worden alle tekens, die voorheen inverse waren geschreven, "omgedraaid" in de normale toestand.

Het eerste verschil met de FLASH-instructie is de eenmalige verandering van de toestand, terwijl FLASH de betreffende tekst voortdurend laat knipperen met de gegeven frequentie. Het tweede verschil met de FLASH-instructie is de werking in een gedefinieerd deel van het beeldscherm. Terwijl FLASH tekens van een bepaalde kleur laat knipperen, differentieert INV niet naar kleur maar naar plaats.

Natuurlijk kunt u met INV een eigen FLASH in elkaar knutselen die alleen werkt op een bepaald deel van het beeldscherm en geen onderscheid maakt naar kleur. Een mankement kan zich voordoen wanneer u de achtergrond wisselt tussen twee INV-instructies die betrekking hebben op hetzelfde deel van het tekstvenster. Het kan dan voorkomen, dat ergens in dat gebied opeens de oude achtergrondkleur verschijnt. U moet hier steeds op letten. Wanneer en of deze fout wordt gecorrigeerd, is nog niet bekend.

Voorbeeld 1:

Voeg de volgende BASIC-regels toe aan het MOVE-voorbeeld van paragraaf 9.4.4.

```
360 INV  1, 0,11, 4
370 INV  1,28,11, 4
380 INV 17, 0,11, 4
390 INV 17,28,11, 4
400 PAUSE 1
410 GOTO 360
```

Voorbeeld 2:

```
100 REM **********************
110 REM **                  **
120 REM ** INV-VOORBEELD 2  **
130 REM **                  **
140 REM **********************
150 REM
160 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
```

```
170 POKE 53248 + 33,12 : REM ACHTERGRONDKLEUR = GRIJS
180 FILL  2,10,10,10,127, 7 : REM = CHR$(191)
190 FILL 14,10,10,10,102,14 : REM = CHR$(166)
200 FILL  2,25,10,10,105,13 : REM = CHR$(169)
210 FILL 14,25,10,10, 97, 2 : REM = CHR$(161)
220 INV 2,10,10,22 : REM EERSTE TWEE
230 INV 2,25,10,22 : REM LAATSTE TWEE
240 FOR X=1 TO 80 : NEXT X : REM KORTE WACHTLUS
250 GOTO 220 : REM BLITSEN
```

Een korte opmerking bij het tweede voorbeeld: in regel 240 werd een wacht-lus ingevoerd. Dit had natuurlijk ook met PAUSE gekund. Maar de duur van PAUSE kan alleen in eenheden van een seconde worden bepaald, zodat deze instructie beter voor langere wachttijden kan worden gebruikt.

9.4.6 Testopdrachten

1. Veronderstel dat u het gebied, begrensd door de regels 5 en 10 en de kolommen 4 en 9, wilt kopiëren op het gebied vanaf regel 8 en kolom 11. Welke schrijfwijze kiest u?

   A) MOVE 5,10, 4, 9, 8,11
   B) MOVE 5, 4,10, 9, 8,11
   C) MOVE 5, 4, 6, 6, 8,11
   D) MOVE 4, 5, 6, 6,11, 8

2. Hoe wordt een deelgebied van het beeldscherm gevuld met het teken A in de rode kleur?

   A) FCHR 5,5,10,10, 1
   B) FILL 5, 5,10,10, 1, 2
   C) FCHR 5,5,10,10, 1 : FCOL 5,5,10,10,2
   D) FILL 5, 5,10,10,65, 2

3. Wat gebeurt er bij de volgende instructie: INV 0,0,0,0?

A) De instructie wordt genegeerd
B) Het teken in de linker bovenhoek wordt geïnverteerd
C) De foutmelding BAD MODE wordt gestuurd
D) De foutmelding ILLEGAL QUANTITY ERROR verschijnt

9.5 Beeldscherm scrollen

Vervolgens worden acht erg bruikbare instructies beschreven die betrekking hebben op het scrollen van het beeldscherm. Deze instructies zijn deels gelijk. Er bestaan 4 basisinstructies (LEFT, RIGHT, UP, DOWN) die telkens de richting aangeven, waarin moet worden verschoven. (dus: naar links, naar rechts, naar boven, naar beneden). Voeg nu bij deze zogenaamde primaire instructies een andere letter als secundaire instructie die de wijze van scrollen bepaalt. Daarvan zijn er 2.

(W)rap around
- W = het cyclisch scrollen van het beeldscherm

(B)lank
- B = het gewoon scrollen van het beeldscherm

Een instructie zou er zo kunnen uitzien: LEFTW 0,0,10,10. Geen spatie tussen de het primaire en het secundaire deel van de instructie!

Cyclisch scrollen is het verschijnsel waarbij die delen, die aan de ene kant uit het venster worden weggeschoven, aan de andere kant weer verschijnen, zodat het beeld steeds rondgaat. Dit effect is vooral bekend van actie-spelen. Bij normaal scrollen 'loopt' het beeld van het scherm af en als het niet met nieuwe tekens wordt aangevuld, wordt het venster leeg. Wanneer u de instructies uitprobeert met grafische voorstellingen zult u merken dat deze alleen werken in de tekstmodus. Als u ook bij graphics vergelijkbare mogelijkheden wilt hebben, moet u een graphicspakket aangeschaffen (bijvoorbeeld SUPERGRAFIEK 64), dat van deze en andere speciale grafische mogelijkheden is voorzien.

9.5.1 LEFT

Format	LEFTW r, k, b, h
Parameter	r : regel (0-24) en
	k : kolom (0-39) van de linker bovenhoek van het veld
	b : breedte van het veld
	h : hoogte van het veld
Voorbeeld	LEFTW 5,7,3,4
Functie	Het naar links scrollen van een deel van het beeldscherm

Met LEFT heeft u een instructie die een rechthoek op het beeldscherm naar links verschuift. Daarbij zijn de beide eerste parameters weer de absolute begincoördinaten van de rechthoek. Ook de andere parameters hebben de betekenis die we al eerder in dit hoofdstuk hebben leren kennen. Dit in tegenstelling tot uitspraken die in het Simon's BASIC Handboek werden gedaan.

```
100 REM *********************
110 REM **                 **
120 REM **  LEFT-VOORBEELD **
130 REM **                 **
140 REM *********************
150 REM
160 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
170 A$ = "CBM 64 --- SIMON'S BASIC///" : REM TEKST DEFINIEREN
180 PRINT AT(7,12) A$ : REM PLAATST TEKST
190 FOR X=1 TO 216 : REM OPTELLEN
200 LEFTW 12,7,LEN(A$),1:REM LEN(A$) IS DE LENGTE VAN A$
210 FOR Y=X TO 100 : NEXT Y : REM VARIABELE WACHTLUS DIE SNELLER
    WORDT
220 NEXT X
230 FOR X=216 TO 1 STEP -1 : REM AFTELLEN
240 LEFTW 12,7,LEN(A$),1:REM LEN(A$) IS DE LENGTE VAN A$
250 FOR Y=X TO 100 : NEXT Y : REM VARIABELE WACHTLUS DIE LANGZAMER
    WORDT
260 NEXT X
270 PRINT : REM LEGE REGEL
280 CENTRE "OVERTUIGD?"
```

Nu wordt het duidelijk, waarom er mensen zijn die dwepen met deze scrolling-instructies. In het voorbeeld wordt een lopende tekst geprogrammeerd die steeds sneller gaat lopen en daarna weer in snelheid afneemt. De elementen die dit effect teweeg brengen, zijn de wachtlussen in de regels 210 en 250. De duur van een wachtlus wordt bepaald door de actuele waarde voor X, die door de FOR...NEXT-lus voortdurend wordt verhoogd (regels 190-220) en in de tweede lus vanaf regel 230 verlaagd. Hier werd de secundaire instructie W gebruikt, zodat niets van de lopende tekst verloren gaat.

9.5.2 RIGHT

Format	RIGHTW r, k, b, h
Parameter	r : regel (0-24) en
	k : kolom (0-39) van de linker bovenhoek van het veld
	b : breedte van het veld
	h : hoogte van het veld
Voorbeeld	RIGHTW 4, 5, 6, 7
Functie	Een deel van het beeldscherm naar rechts scrollen.

RIGHT is de analoge tegenspeler van de LEFT-instructie. Hij verschuift het deel van het beeldscherm, dat met de bekende parameters is gedefinieerd precies in de tegenovergestelde richting. Verder functioneert RIGHT precies hetzelfde. Maar ook hier treedt weer een vervelende onregelmatigheid op. De fout kan bijzonder goed bij een ononderbroken verschuiving naar rechts worden waargenomen. Tijdens het verloop van een instructie kan er een kort knipperen ontstaan van verschillende tekens aan de linker rand van het te verschuiven deel van het beeldscherm. Deze tekens verdwijnen echter na beëindiging van de RIGHT-instructie weer van het scherm. Dit is dus een schoonheidsfoutje, dat misschien in een latere versie kan worden opgeheven. We willen niet te lang bij deze toelichting stilstaan, omdat die immers al bij de instructie LEFT werd gegeven.

Voorbeelden:

```
100 REM ************************
110 REM **                    **
120 REM **  RIGHT-VOORBEELD   **
130 REM **                    **
140 REM ************************
150 REM
160 PRINT CHR$(147): REM BEELDSCHERM WISSEN
170 FOR X=0 TO 39
180 Y = 12 * SIN(X/6) + 12
190 PRINT AT(X,Y) "*": REM SINUSCURVE TEKENEN
200 NEXT X: REM 40 WAARDEN
210 A$ = "------ NOU, HOE VINDT U DIT? ----": REM SCHRIFT DEFINIEREN
220 FOR X=40 TO 1 STEP -1
230 RIGHTB 0,0,40,25 : REM NAAR RECHTS VERSCHUIVEN
240 PRINT AT(0, 9) MID$(A$,X,1): REM TEKST IN BEELD SCROLLEN
250 REM MID$(A$,X,1) SCHRIJFT DE X-TE LETTER VAN DE STRING A$
260 PRINT AT(0,11) ">"
270 PRINT AT(0,12) "<"
280 PRINT AT(0,13) ">": REM TEKENS INSCHUIVEN
290 NEXT X : REM 40 KEER
```

Met dit mooie voorbeeld wordt aangegeven, hoe u iets in een scrollende tekst kunt verpakken. U kunt bijvoorbeeld ook een lichtkrant maken, die veel langer is dan een beeldschermregel. In het bovenstaande programma wordt dit bereikt doordat na iedere verschuiving naar rechts een letter van de te schrijven tekst wordt ingevoerd op de vrijgekomen regelpositie van het zojuist verschoven tekstdeel. In het voorbeeld wordt de tekst, die in het geheugen is opgeslagen, letter voor letter opgeschoven. Dit kan worden bereikt door de instructie MID$ van het originele BASIC. Deze instructie haalt een deel midden uit een string. De te behandelen string of stringvariabele staat op de eerste plaats van het haakje, dat op de instructie volgt: MID$(str$,a,l). Het begin van het deel, dat eruit gehaald moet worden, wordt gedefinieerd door a, de lengte van het fragment bepaalt 1. Nadere informatie geeft uw User Manual en hoofdstuk 8 over stringoperaties.

Nu volgt een gecombineerd RIGHT-LEFT-voorbeeld:

```
100 REM ***************************
110 REM **                       **
120 REM **  RIGHT/LEFT-VOORBEELD **
130 REM **                       **
140 REM ***************************
150 REM
160 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
170 FOR X=0 TO 39
```

```
180 Y = 12 * SIN(X/3) + 12: REM WAARDEN BEREKENEN
190 PRINT AT(X,Y) "*" : REM SINUSCURVE TEKENEN
200 NEXT X : REM*40 WAARDEN
210 FOR Y=1 TO 5
220 FOR X=1 TO 40
230 LEFTW 0, 0,20,25 :REM LINKER DEEL
240 RIGHTW 0,20,20,25 :REM RECHTER DEEL
250 NEXT X : REM 40 MAAL IN TEGENGESTELDE RICHTING VERPLAATSEN
260 FOR X=1 TO 40
270 RIGHTW 0, 0,20,25 :REM RECHTER DEEL
280 LEFTW 0,20,20,25 :REM LINKER DEEL
290 NEXT X : REM 40 MAAL IN TEGENGESTELDE RICHTING VERPLAATSEN
300 NEXT Y : REM 5 MAAL HEEN EN WEER
310 WAIT 198,255 : REM WACHTEN OP EEN TOETSAANSLAG
```

Een opmerking bij dit voorbeeld. In regel 310 verschijnt een nieuwe instructie die onduidelijk is. Deze wordt in het User Manual verklaard, maar let u op het volgende. De functie WAIT 198,255 wacht op uw toetsindruk. Deze wordt daarna met GET A$ opgevraagd. De regel

```
10 WAIT 198,255 : GET A $
```

vervangt dus de regel

```
10 GET A$ : IF A$ = "" THEN GOTO 10
```

Met POKE 198,0 kan een teken, dat toevallig in de toetsenbuffer staat, worden gewist. De geheugenplaats 198 geeft het aantal ingedrukte toetsen aan, dus het aantal tekens, dat zich op de buffer van het toetsenbord bevindt (maximaal 10).

9.5.3 UP

Format	UPW r, k, b, h
Parameter	r : regel (0-24) en
	k : kolom (0-39) van de linker bovenhoek van het veld

	b : breedte van het veld
	h : hoogte van het veld
Voorbeeld	UPW 10, 10, 5, 6
Functie	Een deel van het tekstvenster naar boven laten scrollen

Eigenlijk is er geen nadere verklaring noodzakelijk als u al met RIGHT en LEFT hebt gewerkt. Daarom volgt hier alleen nog een voorbeeld:

```
100 REM ******************
110 REM **              **
120 REM **  UP-VOORBEELD **
130 REM **              **
140 REM ******************
150 REM
160 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
170 PRINT "MET RUN/STOP"
180 PRINT "ONDERBREKEN"
190 FOR X=0 TO 11
200 PRINT AT(X+15,X) CHR$(109) : REM EERSTE DIAGONAAL TEKENEN
210 NEXT X
220 FOR X=12 TO 23
230 PRINT AT(23-X+15,X) CHR$(110) :REM TWEEDE DIAGONAAL TEKENEN
240 NEXT X
250 PAUSE 2
260 UPW 0,15,12,24 : REM NAAR BOVEN SCROLLEN
270 GOTO 260
```

Het is niet erg, wanneer u in het bovenstaande voorbeeld het berekenen van de cursorpositie in het AT-statement niet begrijpt. Voor de liefhebbers wordt dit hier uitgelegd. In de eerste lus (regel 190-210) wordt de eerste diagonaal van linksboven naar rechtsonder punt voor punt berekend. Daarbij ontstaat de waarde van de kolom (eerste waarde tussen de AT-haakjes) door het hoger worden van de X-waarde inclusief een constante 15, die de punten gewoon met 15 kolommen verschuift naar het midden van het beeldscherm. De regelwaarde wordt met de X-waarden hoger, wat wil zeggen, dat in iedere regel precies een punt komt. Op eenzelfde manier wordt nu de teruglopende diagonaal (de regels 220-240) berekend. Hier daalt weliswaar de waarde van de kolom met de stijgende X, omdat met de stijgende X gestadig meer van 23 wordt afgetrokken.

9.5.4 DOWN

Format	DOWN r, k, b, h
Parameter	r : regel (0-24) en
	k : kolom (0-39) van de linker bovenhoek van het veld
	h : hoogte en
	b : breedte van het veld.
Voorbeeld	DOWN 1, 0, 10, 15
Functie	Het beeldschermgebied naar beneden verschuiven (scrollen).

Hiermee zijn we bij de laatste van de groep beeldscherm-scrollers aangekomen, namelijk de instructie DOWN. Hiermee heeft de gebruiker alle mogelijke variaties in de hand. Maar ook hier is nog een klein probleem. Zoals u zich waarschijnlijk herinnert, bestond er bij de RIGHT-instructie een probleem. Daar knipperden diverse tekens aan de rand. Hetzelfde gebeurt ook bij DOWN. Aan de bovenste rand wordt het beeld vaak gestoord door een serie knipperende tekens (bijvoorbeeld apestaarten). U ziet dat ook bij het Motocross-voorbeeld. Net als bij RIGHT zijn deze storingen na beëindiging van de instructie niet meer zichtbaar. Het is ook mogelijk, dat dit effect te maken heeft met de videocontroller van de computer, een hardware-fout. Desondanks geven we u toch dit bijzonder interessante voorbeeld van de mogelijkheden van de DOWN-instructie. Het is nu aan u, om het programma te veranderen en toe te passen. Dat is immers de bedoeling van de voorbeelden.

Voorbeeld:

```
100 REM ********************
110 REM **                **
120 REM **  DOWN-VOORBEELD **
130 REM **                **
140 REM ********************
150 REM
160 REM    ****************
170 REM    **            **
180 REM    **  MOTOCROSS **
```

```
190 REM   **               **
200 REM   *****************
210 REM
220 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
230 POKE 53248 + 33,13 : REM ACHTERGRONDKLEUR = LICHTGROEN
240 X = 10 : REM BEGIN STRAATPOSITIE
250 RV$ = CHR$( 18) : REM RVS ON
260 SC$ = CHR$(144) : REM ZWART
270 GR$ = CHR$(152) : REM GRIJS
280 ST$ = RV$ + SC$ + "!" + GR$ + "     !     " + SC$ + "!"
290 REM STRAAT: ZWARTE RAND - GRIJS WEGDEK (10 SPATIES) - ZWARTE RAND
300 X = X + RND(1)*2 - 1 : REM 0,1,-1 OPTELLEN VOOR TOEVALLIG
    ONTSTANE BOCHTEN
310 IF X < 0 THEN X=0
320 IF X > 38-LEN(ST$) THEN X = 38-LEN(ST$)
330 REM VERKEERDE WAARDEN VERHINDEREN
340 DOWNB 0,0,40,25 : REM STRAAT NAAR BENEDEN VERPLAATSEN
350 PRINT AT(X,0) ST$; : REM STREPEN OP DE WEG ZETTEN
360 GOTO 300
```

9.5.5 Testopdrachten

1. Wanneer u het deel van het beeldscherm, met de regels 6 tot 10 en de kolommen 5 tot 13 naar boven wilt verschuiven met een teken (zonder beeldcirculatie), welke schrijfwijze moet u dan kiezen?

   A) UPW 6,10,5,13
   B) UPD 6,10,5,13
   C) UP D 6,5,9,5
   D) UPD 6,5,9,5
   E) UPW 6,5,9,5

2. Welke combinatie van instructies moet u kiezen, wanneer u het gehele beeldscherm diagonaal naar rechts beneden wilt verschuiven?

   A) RIGHTW 0,0,40,25 : DOWNW 0,0,40,25
   B) LEFTW 0,0,40,25 : UPW 0,0,40,25
   C) is niet mogelijk

D) DOWNW 0,0,40,25 : RIGHTW 0,0,40,25
E) geen enkele van deze mogelijkheden

10 INPUT-CONTROLE

10.1 FETCH

Format	FETCH "CRTL",In,var
Parameter	CRTL - controleteken
	In - maximale lengte van de invoer
	var - willekeurig variabele-type
Functie	gecontroleerde input
Voorbeeld	10 FETCH"12345",1,X$
	verwacht de invoer van een van de getallen '12345' en slaat ze op in X$
Opmerking	De invoer moet door middel van RETURN worden afgesloten

Door FETCH heeft u de beschikking over een instructie, die alle nadelen van een INPUT-instructie ondervangt. De controle van de input van een programma is belangrijk, omdat de gebruiker anders door een verkeerde invoer het programma "ten val kan brengen". Wanneer bijvoorbeeld een waarde van 5 cijfers in een string moet worden ingevoerd, is de instructie 'INPUT A' niet aan te bevelen. Aan de ene kant kan de gebruiker meer dan 5 tekens invoeren, maar aan de andere kant kan hij zelfs door de invoer van een letter een fout veroorzaken. Met de instructie FETCH wordt de invoer van een waarde van 5 cijfers als volgt uitgevoerd:

```
FETCH"0123456789",5,A$
```

Nu worden de tekens 0 tot 9 geaccepteerd en in A$ bewaard. In plaats van de geaccepteerde tekens kunnen controle-tekens worden ingevoerd. Daarmee wordt een bepaald deel van mogelijk in te voeren tekens afgeschermd:

CHR$(19) - alleen hoofdletters (CHR$(65) tot CHR$(90))

CHR$(17) - getallen en speciale tekens (CHR$(32) tot CHR$(64))

CHR$(29) - alleen hoofdletters met en zonder SHIFT

Wanneer bijvoorbeeld een string van 12 ~~cijfers~~ tekens (A$) moet worden ingevoerd, die alleen uit hoofdletters mag bestaan, geldt de volgende FETCH-instructie:

```
FETCH CHR$(19),12,A$
```

Tijdens de invoer kan het laatste teken worden gewist door middel van de toets INST/DEL. Wanneer echter alle benodigde tekens zijn ingevoerd, wordt alleen RETURN nog geaccepteerd en INST/DEL geblokkeerd. Een ander voorbeeld van het gebruik van de instructie FETCH is het volgende. Het komt vaak voor, dat in een programma een antwoord in de vorm van ja of nee wordt gevraagd. Met de GET-instructie werd dit als volgt opgelost:

```
10 PRINT "PROGRAMMA BEEINDIGEN (J/N)?"
20 GET X$:IF X$<>"J" AND X$<>"N" THEN 20
30 IF X$="J" THEN END
40 ..........
```

Met de instructie FETCH gaat het beter:

```
10 PRINT "PROGRAMMA BEEINDIGEN (J/N)?"
20 FETCH "JN",1,X$
30 IF X$="J" THEN END
40 ..........
```

Anders dan met GET moet de invoer met FETCH afgesloten worden met een RETURN.

10.2 INKEY

Format INKEY

Parameter	geen
Functie	Het uitlezen van de functietoetsen
Voorbeeld	A=INKEY
	Slaat het nummer van de bediende functietoets op in de variabele A

Met INKEY kunt u de code van de aangeslagen functietoetsen krijgen (1-16). Daar de Commodore 64 volgens standaard met 8 functietoetsen is uitgerust, worden de overige toetsen met de Commodore-toets gestuurd. Meer hierover vindt u in hoofdstuk 2. INKEY slaat dus de nummers van de bediende functietoetsen op in een van te voren bepaalde variabele. Wanneer er geen toets wordt ingedrukt, wordt de waarde 0 aan de variabele toegekend. Het volgende programma wacht op de bediening van een functietoets en geeft ze dan aan:

```
10 X=INKEY:IF X=0 THEN 10
20 PRINT "FUNCTIETOETS NR."X
30 GOTO 10
```

Dit programma kan alleen door de toets RUN/STOP worden onderbroken. De werkwijze is die welke ook vaak wordt toegepast om in standaard-BASIC een ASCII-waarde in te lezen, maar in plaats van GET gebruiken we INKEY. Omdat alleen aan de normale 8 functietoetsen een ASCII-waarde wordt toegekend (133-140), kunnen de 8 extra toetsen alleen door INKEY worden ingelezen. Het inlezen van de functietoetsen kan bijvoorbeeld noodzakelijk zijn bij het kiezen uit een menu als:

```
Kies de gewenste functie:
--------------------------------
F1 - adressen invoeren
F2 - adressen wissen
F3 - adressen opgeven
F4 - adressen laden
F5 - adressen opslaan
```

De regels, die deze keuze sturen, kunnen er zo uitzien:

```
100 A=INKEY:IF A=0 THEN 100
110 CGOTO (A*10 + 110)
120 CALL INVOEREN
130 CALL WISSEN
```

```
140 CALL INDEX
150 CALL LADEN
160 CALL SAVEN
```

Verder kunnen de gebruikers van een programma met behulp van de functietoetsen beslissingen nemen, bijvoorbeeld:

UITVOER OP PRINTER (F1) OF BEELDSCHERM

De programmaregels, die hierbij gestuurd moeten worden, zijn:

```
100 A=INKEY:IF A=0 THEN 100
110 IF A=1 THEN CALL PRINTER
```

10.3 RESET

Format	RESET rn
Parameter	rn - nummer van de DATA-regel, waarop de DATA-pointer moet worden gezet
Functie	De DATA-pointer op de gegevens zetten die met READ gelezen worden
Voorbeeld	RESET 100 De DATA-regels vanaf 100 worden door de volgende READ gelezen

Om een groot aantal constanten in het geheugen op te slaan, gebruikt men in BASIC DATA-regels en de instructie READ. Elke keer als de READ-instructie actief is, wordt de wijzer op de volgende DATA-waarde gezet, zodat elke READ-instructie de volgende waarde leest. Wanneer u probeert meer DATA te lezen dan ;er in het programma aanwezig zijn, verschijnt de foutmelding "OUT OF DATA ERROR". Met de instructie RESTORE van standaard-BASIC wordt de pointer weer op de eerste DATA-waarde gezet. De volgende twee programmasegmenten zullen dit duidelijk maken.

```
200 DATA WILLEM,ARJEN,RIEMKE,RINDERT*
300 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,*
400 DATA 102,103,105,108,123,120,142,189,*
500 DATA NEDERLAND,BELGIE,LUXEMBURG,DUITSLAND,*
```

Dit zijn DATA-regels die qua inhoud van elkaar verschillen. De gegevens worden gelezen en in variabelen opgeslagen. Wanneer in het verloop van een programma met standaard-BASIC alleen de landen moeten worden ingelezen, moet u door middel van RESTORE weer terugkeren naar de positie van het eerste DATA-statement. Vervolgens moeten de eerste DATA-waarden worden overgelezen. Het aantal hiervan moet dus bekend zijn. Dat is omslachtig en tijdrovend. Met Simon's BASIC gaat het eenvoudiger. Voer alleen maar de instructie RESET 130 in en de DATA-pointer wordt op de eerste waarde in regel 130 geplaatst. Wanneer deze door RESET aangegeven regel niet aanwezig is, of niet gedefinieerd is met DATA-waarden, wordt er automatisch overgeschakeld naar de volgende regel. Een voorbeeld voor het positioneren van de pointer met RESET:

```
10 PRINT"KIEST U (1-4):"
20 PRINT"--------------"
30 PRINT"1 - NAAM"
40 PRINT"2 - HEX.GETALLEN"
50 PRINT"3 - GETALLEN"
60 PRINT"4 - LANDEN"
70 FETCH"1234",1,A$
80 IF A$="1" THEN RESET 200
90 IF A$="2" THEN RESET 300
100 IF A$="3" THEN RESET 400
110 IF A$="4" THEN RESET 500
120 READ A$
130 IF A$="*" THEN END
140 PRINT A$
150 GOTO 120
```

10.4 Testopdrachten

1. Welke tekenreeks werd niet ingevoerd met de instructie 'FETCH"0123456789ABCDEF$",3,A$'?

A) $87
B) $FF
C) A$A
D) C $

2. Welke regels werden door de volgende instructie geaccepteerd?
FETCH CHR$(19),3,X$

A) alleen cijfers en speciale tekens
B) alleen hoofdletters met en zonder SHIFT
C) allleen hoofdletters

3. Hoeveel functietoetsen kunnen met behulp van de instructie INKEY worden gecontroleerd?

A) 8
B) 16
C) 15
D) 7

4. Welke instructie kan de DATA-wijzer op willekeurige regels plaatsen?

A) RESTORE
B) RESET
C) CRESTORE

11 IN- EN UITVOERINSTRUCTIES

11.1 Diskette-instructies

11.1.1 DISK

Format	DISK "aanwijzing"
Parameter	aanwijzing - diskdrive-instructie
Functie	vereenvoudigde communicatie met de diskdrive
Voorbeeld	DISK "N:TESTDISK,MN" formateert een diskette

De gebruikelijke omweg bij het overdragen van een diskdrive-instructie vervalt met de instructie DISK. Zonder DISK moet u om bijvoorbeeld een diskette te formateren de volgende instructie geven:

```
OPEN1,8,15,"N:TESTDISK,MN"
CLOSE1
```

Met de DISK-instructie gaat dat gemakkelijker. Het floppy-instructiekanaal hoeft niet meer geopend en gesloten te worden. Deze taak neemt de instructie DISK over. Voor wie niet met de diskdrive-instructies vertrouwd zijn, geven we nu een beschrijving van de belangrijkste daarvan:

NEW	diskette formateren
Format	DISK "N:naam,id"
Parameter	naam - naam van de diskette (max. 16 tekens) id - kenmerk van de diskette (2 tekens)
Voorbeeld	DISK "N:SIMON'S BASIC,NL"
SCRATCH	files uitwissen
Format	DISK "S:filename1, filename2,..."

Parameter	filename1 - 1e uit te wissen file filename2 - 2e uit te wissen file
Voorbeeld	DISK "S:TESTPROGRAMMA"
RENAME	files herbenoemen
Format	DISK "R:nieuwe file=oude file"
Parameter	nieuwe file - nieuwe filenaam oude file - filenaam tot nu toe
Voorbeeld	DISK "R:INFO=INFO.TEST"
INITIALISE	initialiseren van de diskette
Format	DISK "I"
Parameter	geen
VALIDATE	overbodige blokken uitwissen
Format	DISK "V"
Parameter	geen

Wie meer van de veelzijdige floppy-instructies wil weten, kunnen wij "HET GROTE FLOPPYBOEK" aanbevelen van DATA-BECKER/NEDERLANDS*.

11.1.2 DIR

Format	DIR"$" DIR"$:string*" DIR"$:??string" DIR"$:*=ft"
Parameter	string - filenaam of deel van een filenaam

? * - Joker (zie tekst)

ft - filetype (P=PRG, S=SEQ, R=REL, U=USR)

Functie — een inhoudsopgave van de diskette weergeven zonder dat een "staand" BASIC-programma gewist wordt

Voorbeeld — DIR"$" laat de hele directory zien

Deze instructie is, zoals u ziet, zeer veelzijdig. Vier verschillende schrijfwijzen, die zelfs nog gemengd kunnen worden. De eerste vorm is de eenvoudigste. Voor het aanwijzen van de directory van de diskette moet u de instructie DIR"$" gebruiken. In tegenstelling tot het standaard-BASIC blijft hier het BASIC-programma behouden. Maar er zijn lastigere formats. Met een ster kunt u van de directory een selectieve list maken. Als u bijvoorbeeld de instructie 'DIR "$:A*" opgeeft, worden alle files van de diskette die met de hoofdletter A beginnen, aangegeven. De ster geeft dus aan dat het niet gespecificeerde gedeelte van een gevonden filenaam er niet toe doet. Nog een voorbeeld: 'DIR "$:TEST*" toont alle files, die met de letters 'TEST' beginnen (TEST.1, TEST.NIEUW, TESTPROGRAMMA enzovoort). Het vraagteken vervangt die posities in de filenaam, die voor het maken van een selectieve lijst overgeslagen moeten worden. Ze mogen ook voor of midden in de filenaam zitten. Een voorbeeld: 'DIR "$:???.01" toont alle filenamen van 6 posities, waarvan de laatste drie tekens '.01' zijn. Of 'DIR "$:TEST.??" toont alle filenamen van 7 posities, waarvan de eerste 5 tekens 'TEST.' zijn. Het vraagteken kan ook met de ster worden gecombineerd. Op deze manier toont bijvoorbeeld de instructie 'DIR "$:???.01*" alle filenamen van de diskette, waarvan het 4e, 5e en 6e teken met '.01' overeenkomen. Verder kunnen files ook op type-overeenkomst geselecteerd worden. 'DIR $:*=p"' bijvoorbeeld geeft alle programmafiles, die op de diskette staan. Ook deze vorm kan met de andere gecombineerd worden. De vraag, wat de instructie 'DIR"$:???.18*=p' nu als output heeft, laat ik over aan die lezers, die mij tot nu toe zonder problemen konden volgen. Proberen is altijd beter dan het bestuderen. Test u deze verschillende schrijfwijzen van de DIR-instructie daarom net zolang, tot u ze onder de knie hebt. De instructie DIR kunt u ook in een programma gebruiken. Dan is het zinvol eerst het beeldscherm vrij te maken en het programma na het uitvoeren van de directory met een toetsaanslag voort te zetten. Het volgende programma geeft een voorbeeld.

```
100 PRINT"DIRECTORY ZIEN (J/N)?"
110 REPEAT
120 ::GET X$
130 UNTIL X$="J" OR X$="N"
140 IF X$="N" THEN CALL VERDER
```

```
150 ::PRINT CHR$(147):REM BEELDSCHERM WISSEN
160 ::DIR"$":REM DIRECTORY LISTEN
170 ::PRINT:PRINT"VERDER MET RETURN"
180 ::REPEAT
190 :::GET X$
200 ::UNTIL X$=CHR$(13)
210 PROC VERDER
```

Interessant is het feit, dat een reeds bestaand programma niet zoals bij het normale 'LOAD"$",8' wordt gewist. Hoe is dat te verklaren? Omdat de instructie 'LOAD' zonder secundair adres 1 altijd vanaf het begin van het BASIC-programma-geheugen laadt, wordt een daar aanwezig BASIC-programma overschreven door het nieuwe. Bij de instructie DIR wordt de directory als data byte voor byte gelezen en op het beeldscherm geschreven. De ingelezen directory wordt dus niet in het hoofdgeheugen van de CBM 64 opgeslagen, maar alleen op het beeldscherm getoond.

11.1.3 SCRSV

Format	SCRSV lfn,aa,sa,"naam,S,W"
Parameter	lfn - logisch filenummer
	aa - apparaatadres (1=cassette 8=floppy)
	sa - secundair adres
	naam - bestandsnaam voor de beeldscherminformatie
Functie	vastleggen van beeldscherminformatie in de lage resolutie of tekstmodus
Voorbeeld	SCRSV 1,8,2,"TEST.SCRSV,S,W"

Het LOW-RESOLUTION-beeldscherm, dus het beeldscherm zoals dat na het inschakelen is, kan met deze instructie op diskette of cassette vastgelegd worden. Er kan grafiek met lage resolutie en tekst met de instructie SCRSV op diskette als opeenvolgende data vastgelegd worden. Dat is bijvoorbeeld bij het werken met beeldschermmaskers zinvol. Een beeldschermmasker is een soort invulkaart, dus een

techniek om gegevens op te nemen. U hoeft dit beeldschermmasker maar een keer in het programma te verwerken. Daarna slaat u het op in het geheugen en roept u het weer op als u het nodig hebt. De parameters van deze instructie komen overeen met de parameters om een sequentieel bestand te openen. Dat betekent: er mag bijvoorbeeld geen file met het overeenkomende logische filenummer al open zijn, omdat anders de foutmelding "FILE OPEN ERROR" te voorschijn komt.

Een voorbeeld:

```
100 REM ******************
110 REM   SCRSV-VOORBEELD
120 REM ******************
130 PRINTCHR$(147)
140 PRINT"NAAM          :....................."
150 PRINT"VOORNAAM      :....................."
160 PRINT"STRAAT        :....................."
170 PRINT"PLAATS        :....................."
180 PRINT"TELEFOON      :....................."
190 SCRSV 1,8,2,"TEST.SCRSV,S,W"
200 GOTO 200
```

De instructie SCRSV kan zowel in de programma- als in de direkt-modus gebruikt worden, maar in de direkt-modus wordt natuurlijk ook de SCRSV-instructie, die nog op het beeldscherm staat, als bestanddeel van het tekstbeeldscherm opgeslagen. Helaas kan de grafische informatie in de hoge resolutie niet opgeslagen worden. Deze mogelijkheid wordt wel geboden met het uitbreidingspakket SUPERGRAFIEK 64, dat de volgende voordelen heeft:

1. Een grafiek, waarvan het maken in sommige gevallen zeer veel tijd in beslag kan nemen, (bijv. een driedimensionale grafiek met omvangrijke wiskundige berekeningen) kan in een paar seconden van de diskette gehaald en op het beeldscherm weergegeven worden.

2. De opgeslagen grafiek in hoge resolutie van verschillende grafiekprogramma's voor de CBM 64 zijn tegen elkaar uit te wisselen.

11.1.4 SCRLD

Format	SCRLD lfn,aa,sa,"naam"
Parameters	lfn - logisch filenummer
	aa - apparaatadres (1=cassette 2=disk)
	sa - secundair adres
	naam - bestandsnaam van beeldscherminformatie
Functie	laden van met SCRSV opgeslagen beeldscherminformatie in de lage resolutie
Voorbeeld	SCRLD 1,8,2,"TEST.SCRLD"
	toont beeldscherminformatie met de naam "TEST.SCRLD"

Beeldscherminformatie, die met SCRSV opgeslagen werd, wordt opnieuw geladen en weergegeven. Het beeldscherm bevat die grafische gegevens, die voor het opslaan aanwezig waren. U hebt dus de mogelijkheid ook de beeldscherminformatie van een ander programma van diskette of cassette te halen. Omdat in het programma slechts een enkele instructie voldoende is en het laden vanaf diskette relatief snel verloopt, krijgt u de gewenste beeldschermweergave met een verbluffend gemak. In het programma ziet het er als volgt uit:

```
100 REM ******************
110 REM   SCRLD-VOORBEELD
120 REM ******************
130 SCRLD 1,8,2,"TEST.SCRSV"
140 GOTO 140
```

Met de toets 'RUN/STOP' komt u weer uit dit programma. Natuurlijk kunt u de beeldscherminformatie ook weergeven in de direkt-modus, dus zo:

SCRLD"1,8,2,"TEST.SCRSV"

11.1.5 Testopdrachten

1. De file "TEST.SCRSV" moet uitgewist worden.
   Welke instructie is juist?

   A) DISK "S:TEST.SCRSV"
   B) DISK "N;TEST.SCRSV"
   C) DISK "R:TEST.SCRSV"

2. Welke file wordt niet door de volgende instructie getoond?
   DIR"$:????.TE"

   A) VARB.TEST
   B) KA01.TE
   C) SCRL.TE

3. Welke file wordt niet door de volgende instructie getoond?
   DIR "$:??FILE*"

   A) MNFILES
   B) MAFILE.PROGRAMMA
   C) FILEMANAGER

4. Een LOW-RESOLUTION-beeldscherm moet op diskette opgeladen
   worden. Welke instructie is juist?

   A) SCRSAVE 1,8,2,"LOWRES01,S,W"
   B) SCRSV 8,8,3,"LOWRES01",S,W
   C) SCRSV 2,8,2,"LOWRES01"
   D) SCRSV 5,8,3,"LOWRES01,S,W"

11.2 Printinstructies

11.2.1 COPY

Format	COPY
Parameter	geen
Functie	Geeft een hardcopy van grafiek in HI-RES of MULTI COLOUR

De instructie COPY opent en sluit zelfstandig het printerkanaal. Een OPEN 1,4 voorafgaand aan de instructie is dus niet nodig.

De instructie werkt met de volgende printers:

- -CBM VC-1520
- -CBM VC-1525
- -MPS-801 + 803
- -SEIKOSHA GP 100 VC
- -EPSON RX-80, FX-80 met DATA-BECKER-Interface

Omdat het grafisch printen met de CBM VC-1526 anders werd georganiseerd, is de instructie COPY daarvoor niet bruikbaar. Een speciale hardcopy-routine voor de VC-1526 vindt u in SUPERGRAFIEK 64. Deze instructie kunt u zowel in de programma- als in de direkte modus gebruiken. Als u bijvoorbeeld na het tekenen van een grafiek met behulp van een programma een hardcopy wilt maken, voegt u aan dit programma een regel met de instructie COPY toe. U kunt echter ook na afloop van het programma, als de computer zich meldt met 'READY', de instructie COPY gebruiken, ook als de machine dan in de LOW-RESOLUTION (tekstmodus) staat. De beeldscherminformatie in de hoge resolutie hoeft dus niet zichtbaar te zijn om deze als hardcopy op de printer te zetten.

11.2.2 HRDCPY

Format	HRDCPY
Parameter	geen

Functie Hardcopy van de beeldscherminformatie in de lage resolutie

Met deze instructie kunt u de normale beeldscherminformatie, die u op de diskette kunt laden, ook op de printer afdrukken. In sommige programma's is het zinvol om "met een druk op de knop" een afbeelding van de inhoud van het beeldscherm te verkrijgen. Het daarvoor verantwoordelijke programmagedeelte zou er zo uit kunnen zien:

```
100 PRINT "HARDCOPY GEWENST (J/N)?"
110 FETCH "JN" 1,X$
120 IF X$="J" THEN HRDCPY
130 .........
```

Natuurlijk kunt u de instructie ook in de direktmodus invoeren. Deze instructie is ook bruikbaar met CBM VC-1526.

11.2.3 Testopdrachten

1. Welke uitvoer geeft de instructie COPY?

   A) HIGH-RESOLUTION
   B) MULTI-COLOUR
   C) LOW-RESOLUTION

2. Welke uitvoer geeft de instructie HRDCPY?

   A) HIGH-RESOLUTION
   B) MULTI-COLOUR
   C) LOW-RESOLUTION

3. Is het wenselijk, voor gebruik van de instructies COPY en HRDCPY het printkanaal te openen? (OPEN1,4)

   A) ja
   B) nee

12 GRAFIEK

We staan op het punt het hoogtepunt van het programmeren te bereiken, de droom van elke computerbezitter en het ideaal van de programmeerkunst: de grafiek! Pas door het maken van grafische voorstellingen wordt de computer tot computer. Het is zaak om informatie zo sprekend en overtuigend mogelijk weer te geven, de kunst ook de leek de ongelooflijke uitgebreidheid en het nut van de moderne informatieverwerking en niet te vergeten, van de vrijetijdsbesteding, aan te tonen. Samen met de geluidsmogelijkheden van de synthesizer ligt in de grafische mogelijkheden het hoofddoel van de computerkunst. Niet voor niets hebben de technici zich bij het ontwerp vande Commodore 64 geen enkele moeite bespaard, om bij dit apparaat de mogelijkheden open te houden. Het is de taak van de software-specialisten om deze mogelijkheden op een gemakkelijk toegankelijke manier vorm te geven. Daarom mag de grafiek uiteraard in uw Simon's BASIC-boek niet ontbreken, ook niet wanneer het de grafische mogelijkheden (Simon's BASIC is immers een algemene BASIC-uitbreiding) niet volledig benut. Als u zich intensief met de grafiek en de geluidsmogelijkheden van uw computer wilt bezighouden, zult u een speciale grafiekuitbreiding als SUPERGRAFIEK 64 moeten aanschaffen.

Voordat we met de uitleg van de aparte instructies beginnen, moeten we u voor een beter begrip en vanwege een eenduidige terminologie eerst wijzen op de grondbeginselen van de grafiekmogelijkheden in uw CBM 64. Daarbij worden hier alleen die mogelijkheden uitgelegd, die ook in Simon's BASIC aan de orde zijn. Als u zich speciaal met de grafiekverwerking van uw computer wilt bezighouden, bestaat er daarvoor literatuur (bijvoorbeeld 64 INTERN en HET GRAFIEKBOEK)

12.1 Hardwarefaciliteiten

De eerste wetenswaardigheden volgen hier; sprites, karakterset en synthesizer worden in de volgende hoofdstukken behandeld.

Kleuren:

Uw Commodore 64 bezit de mogelijkheid, om 16 kleuren, zowel in de grafiektekening als voor de tekstverwerking te gebruiken. Deze 16 kleuren bezitten steeds een zogenaamde kleurcode, die als een binair getal in de verschillende registers opgeslagen wordt. Als bijvoorbeeld in het 32e register van de Video Interface Chip de waarde 0 gePOKEd wordt, krijgt het buitenste kader van het beeldscherm de kleur zwart.

0 - zwart	8 - oranje
1 - wit	9 - bruin
2 - rood	10 - lichtrood
3 - cyaan	11 - grijs 1
4 - violet	12 - grijs 2
5 - groen	13 - lichtgroen
6 - blauw	14 - lichtblauw
7 - geel	15 - grijs 3

Hoogoplossende grafiek:

Uw computer heeft van huis uit de mogelijkheid twee verschillende grafieksoorten te bedienen: de hoogoplossende grafiek (HGR) en de multicolourgrafiek (MC). De eerste biedt een grafiekbeeld van 320 punten in de x-richting en 200 punten in de y-richting. Dit verschaft u een werkveld van in totaal 64000 punten steeds in gelijke afstand op uw monitor verdeeld. Natuurlijk moet de grafiek net zoals de tekst (zie hoofdstuk 9) opgeslagen zijn. Dat gebeurt in het zogenaamde grafiekgeheugen. Elk punt is apart oproepbaar en wordt in dit grafiekgeheugen door een bit vertegenwoordigd. Een bit is een informatie-eenheid en kan de waarde 1 of 0 aannemen. Steeds 8 bits achter elkaar duidt men aan als een byte. Een byte kan dus twee tot de achtste (2 tot de 8e) = 256 verschillende waarden aannemen. Deze komen tot stand door de verschillende combinaties van gezette (1) en niet-gezette (0) bits. Een byte vertegenwoordigt dus 8 punten op het beeldscherm. Dus zijn er 64000/8 = 8000 bytes (ongeveer 8 kilobyte (8K) nodig, om de gehele inhoud van het beeldscherm van de HGR op te slaan. Deze 8K worden in Simon's BASIC ruimtebesparend in het RAM onder het ROM van het bedrijfssysteem van hexadecimaal $E000 tot $FF40 (decimaal van 57344 tot 65344) afgezet. Als u dus direkt in het beeldschermgeheugen wilt POKEn, kunt u deze adressen gebruiken. Om zulke direkte manipulaties te kunnen uitvoeren, moet u weten hoe nu de opbouw van het grafiekbeeld uit de informatie van het geheugen gebeurt: basis van het beeldschermgeheugen is een 8*8 matrix, die steeds door 8 bytes van het grafiekgeheugen wordt bepaald. Deze lijkt op de matrix van de karakterset voor de tekstmodus. Omdat zo'n matrix dus 8 punten hoog en breed is, passen er in totaal 40 van zulke pakjes in een regel en 25 in een kolom. Dit is als volgt voor te stellen:

```
    Bit   7 6 5 4 3 2 1 0   7 6 5 4 3 2 1 0   ....
Byte 0    . . . . . . . .   . . . . . . . .
Byte 1    . . . . . . . .   . . . . . . . .
Byte 2    . . . . . . . .   . . . .
Byte 3    . . . . . . . .
Byte 4    . . . . . . . .
Byte 5    . . . . . . . .
Byte 6    . . . . . . . .
Byte 7    . . . . . . . .

Byte 320  . . . . . . . .   . . . . . . . . .
Byte 321  . . . . . . . .   . . . .
Byte 322  . . . . . . . .
...
```

Regel voor regel wordt dus uit die pakjes gegenereerd. Belangrijk is dit ook voor het bepalen van de kleur. U heeft 16 kleuren naar keuze, waarmee u de achtergrond (niet-gezette punten) en ook de gezette punten kunt weergeven. Omdat er echter niet genoeg geheugenruimte beschikbaar is om elk van de 64000 punten een eigen kleur te geven, (u zou daarvoor 64000*4 = 256000 bits, dus 32 K RAM nodig hebben), werd steeds een 8*8 pakje als een kleureenheid gekozen. Dat betekent dat u steeds voor zo'n pakje de kleur voor alle niet-gezette (achtergrondkleur) en alle gezette (puntkleur) punten kunt bepalen. Hiervoor is 1Kbyte geheugen nodig, het VIDEORAM. Dit ligt, eveneens buiten het programmageheugen van $C000 tot $C3E7 (decimaal: 49152 - 50151). Het is net zo georganiseerd als het beeldschermgeheugen van de tekstmodus. In elke byte van dit videoram wordt in de bovenste 4 bits de kleur van de punten en in de onderste 4 bits de achtergrondkleur van een 8*8 blok opgeslagen. Als u vanuit BASIC op een bepaalde positie een punt wilt plaatsen, (wat sneller met de betreffende instructie van Simon's BASIC gaat), of alleen wilt testen of het geplaatst is, kunt u het beste de volgende formule voor de berekening van de adressen gebruiken:

```
10 REM X EN Y ZIJN DE BEIDE COORDINATEN V/H PUNT
20 AD=57344:REM BASISADRES V/H GRAFIEKGEHEUGEN
30 OY = 320 * INT(Y/8) + (Y AND 7)
40 OX = 8 * INT(X/8)
50 MA = 2 ^ (7-(X AND 7))
60 AV = AD + OY + OX
70 REM VOORBEELD: PLAATSEN VAN EEN PUNT
80 POKE AV,PEEK(AV) OR MA
```

U kunt ook zonder deze formule leven, omdat er immers al instructies voor het plaatsen in Simon's BASIC zijn. Nu volgt een eenvoudig

voorbeeld van het direkt inPOKEn in het grafiekbeeldscherm om een streeppatroon te maken.

```
10 HIRES 2,1: REM GRAFIEK INSCHAKELEN
20 FOR X=57344 TO 65344
30 POKE X, %11110000 : REM EERSTE 4 BITS PLAATSEN
40 NEXT X
50 WAIT 198,255:REM WACHTEN OP TOETSAANSLAG
```

U ziet, hoe langzaam zo'n BASIC programma is.

Multicolourgrafiek:

Uw computer heeft in de HGR-modus slechts een zeer beperkte kleurmogelijkheid. Om dit door de geheugenruimte bepaalde manco op te heffen, werd aanvullend bij de HGR-modus de zogenaamde multicolourmodus gemaakt. Deze maakt het mogelijk, in een 8*8 blok in plaats van twee, in totaal vier kleuren tegelijkertijd te gebruiken. Dat gaat echter noodzakelijk ten koste van de resolutie. In de multicolourmodus staan telkens twee bits voor een dubbel breed punt op het beeldscherm. De resolutie bedraagt daardoor 160 dubbelbrede punten in de x-richting en 200 punten in de y-richting (160*120). De beide bits van het grafiekgeheugen, die samen een punt voorstellen, geven nu het nummer van een van de vier mogelijke kleurregisters van een blok aan. Met nummer wordt hier niet de kleurcode (0-15) bedoeld, maar de toewijzing van het register, dat de kleurcode bevat. Daarbij staat in elk van de vier kleurregisters de eigenlijke kleurcode van het punt. Aan de vier kleurregisters zijn de volgende geheugenposities toegekend:

Kleurregister	Bit-code	Geheugen
0	00	Register 33 van VIC
1	01	onderste 4 Bits van het video-RAM
2	10	bovenste 4 Bits van het video-RAM
3	11	kleur-RAM

Staat dus bijvoorbeeld in de desbetreffende byte van het grafiekgeheugen een bitpaar 10, dan heeft het punt die kleur, waarvan de code in de bovenste 4 bits van de daarbijhorende byte van het videoram (kleurgeheugen) opgeslagen is. Het 33e register van de VIC is u al als register bekend. Dat bevat ook in de tekstmodus de achtergrondkleur en wordt door POKE 53248+33,k gestuurd, waarbij k de kleurcode voorstelt. Het kleurram is het geheugenbereik dat in de

tekstmodus de kleur van het teken geeft. U kunt voor elk 8*8 blok de kleuren 1-3 definiëren. Het volgende voorbeeld kan u de situatie verduidelijken.

```
10 HIRES 2,1 : MULTI 12,8,6 : REM MULTICOLORGRAFIEK INSCHAKELEN
20 FOR X=57344 TO 65344
30 POKE X, %11001001   : REM 3 EEN-PUNT-LIJNEN MET 3 KLEUREN TEKENEN
40 NEXT X
50 WAIT 198,255:REM WACHTEN OP TOETSAANSLAG
```

Dit voorbeeld tekent, steeds door een spatie (achtergrondkleur = 00) gescheiden, een punt brede strepen op het beeldscherm, waarvan de drie kleuren steeds wisselen.

Tot zover de hardware-beginselen, die u niet echt hoeft te onthouden, maar die u inzicht in de zeer gecompliceerde grafische mogelijkheden van de Commodore 64 moeten bieden om de volgende uiteenzettingen te begrijpen Dat is een goede hulp om later mooie grafiek te maken.

12.2 Grafiekmodus/Kleurbepaling

Coördinatensysteem:

Simon's BASIC verdeelt uw grafiekbeeldscherm in 320*200 punten (van 0 tot 319) in de hoogoplossende grafiek (HGR) en 160*200 punten in de multicolourmodus (0-159); in multicolour is elk punt dus dubbel zo breed als in HGR. Dit heeft speciale vertekeningsefffekten, als men x- en y-coördinaten gelijkwaardig behandelt. Voor elk punt moet u dus een x- en een y-coördinaat opgeven om het te bepalen. Daarbij werd de coördinatenoorsprong (0,0) in de linker bovenhoek van het beeldscherm verplaatst. Dat wil zeggen dat de y-coördinaten van boven naar beneden tellen en de x-coördinaten van links naar rechts. Het punt 319,199 ligt in HGR bijvoorbeeld in de rechter benedenhoek van het beeld. Mocht u een keer een niet-bestaand coördinaat (bijvoorbeeld 400,600) opgeven, dan onstaat er een ILLEGAL QUANTITY ERROR. De kleuren en de kleurmogelijkheid in HGR en multicolour werden al in de paragraaf "grondbeginselen" uitvoerig behandeld. U heeft de mogelijkheid direkt een grafiek te tekenen, terwijl die door de toeschouwer op het beeldscherm bekeken kan worden. U kunt echter ook elk van de onderstaande grafiekinstructies gebruiken, ook als u zich in de tekstmodus bevindt. Dat wil zeggen: u kunt een

grafiek onzichtbaar tekenen en later zichtbaar maken. Om met een klein aantal instructies zoveel mogelijk mogelijkheden te benutten, werd de tekenmodus voor elke tekeninstructie (LIN, REC,...) ingevoerd. Deze bepaalt de manier, hoe een punt, een lijn, een rechthoek, enzovoort op het beeldscherm getekend wordt. Dat betekent dat zo'n element door middel van de tekenmodus een verschillend uiterlijk kan krijgen. De desbetreffende tekenmodus wordt steeds in de te verwerken instructie aangegeven (zie daarvoor de afzonderlijke instructies). We onderscheiden daarbij de variaties van de tekenmodus in multicolour en die in HGR. Het volgende schema verduidelijkt de afzonderlijke functies:

- hoogoplossende grafiek (HGR):

tekenmodus	functie
0	wist een produkt uit
1	tekent een produkt
2	inverteert een produkt (zie beneden)

- multicolourmodus (MC):

tekenmodus	functie
0	wist een produkt uit
1	tekent een produkt met de kleur van het kleurregister 1
2	tekent een produkt met de kleur van het kleurregister 2
3	tekent een produkt met de kleur van het kleurregister 3
4	inverteert een produkt = wisseling van kleurregister: kleurreg. 0 naar kleurreg. 3 kleurreg. 1 naar kleurreg. 2 kleurreg. 2 naar kleurreg. 1 kleurreg. 3 naar kleurreg. 0

Deze tabel heeft natuurlijk een verklaring nodig.

Ten eerste wordt onder produkt een door u bepaalde figuur verstaan, dus punt, lijn, cirkel enzovoort, dat u direkt met een instructie van Simon's BASIC kunt maken. Het inverteren van een produkt betekent hierbij het uitwissen van een punt op de plaats van de te tekenen figuur, waarop al een punt geplaatst was en overeenkomstig omgekeerd de plaatsing daar, waar nog geen punt stond. Met kleurregisters worden die geheugenlocaties bedoeld, die voor het vastleggen van een van de vier mogelijke kleuren van een 8*8 blok bepaald zijn. In een kleurregister wordt dus de betreffende kleurcode (0-15) vastgelegd.

Alle punten moeten met het kleurregister in verband worden gebracht door middel van de twee voor elk punt in het grafiekgeheugen gereserveerde bits. Met kleurregister 0 wordt daarbij de normale achtergrondkleur bedoeld, die immers in de multicolourmodus uit het achtergrondkleur-register 0 afkomstig is. Pas de tekenmodus maakt uw instructies tot een flexibel complex van grafiekinstructies.

12.2.1 HIRES

Format	HIRES pk,ak
Parameter	- pk: puntkleur (0-15)
	- ak: achtergrondkleur (0-15)
Voorbeeld	HIRES 7,8
Functie	Uitwissen van de grafische beeldscherminformatie, inschakelen van de hoogoplossende grafiek (HGR) en bepalen van de punt- en achtergrondkleur

De instructie, die voor u het rijk der grafiek opent, is de eerste in een serie instructies voor het in werking stellen van de grafiekmodus, de bewerking van grafische beelden en de kleurbepaling. Ze vormen de basis van alle grafiekprogrammering. De eerste instructie, HIRES, is voor het omschakelen van de tekst- naar de grafiekmodus. Tegelijk wist hij het grafiekgeheugen, zodat u na de invoer van deze instructie een leeg beeldscherm voor u ziet. Dat is belangrijk om te weten, omdat men vaak tijdens een programma even in de tekstmodus terug springt. Wilt u daarna weer dezelfde grafiek onveranderd laten zien, dan moet er met een andere instructie gewerkt worden, die de grafiek zonder uitwissen inschakelt. Natuurlijk kunt u de HIRES-instructie ook gebruiken, terwijl u de grafiek laat zien. In dat geval wordt er slechts gewist en de kleuren gereset. Bij het wissen van het grafiekbeeld worden alle bits van het grafiekgeheugen op 0 teruggezet en alleen de achtergrondkleur uit het videoram is zichtbaar. Deze en de later gebruikte puntkleur kunt u met beide, op de HIRES-instructie volgende parameters vastleggen: de eerste waarde definieert de kleur (kleurcode) van alle gezette punten van het totale grafiekbeeldscherm, tenzij door een andere kleurinstructie veranderd. Dat wil zeggen, alle punten die nu geplaatst worden, krijgen deze ene kleur, als ze niet door een extra instructie (LOW COL) met een andere kleur getekend

worden. De uiteenzettingen over de puntkleur spelen natuurlijk alleen in HGR een rol. De kleuren van de punten in de multicolourmodus worden extra geplaatst, zodat in MC de in de HIRES-instructie gezette puntkleur genegeerd wordt, evenals de twee parameters van HIRES.

In de tweede waarde geeft u de kleurcode van de achtergrondkleur aan, dat wil zeggen de kleur van alle "niet-gezette" punten in het grafiekvenster. De hier gezette achtergrondkleur heeft echter niets met de door COLOUR te veranderen achtergrondkleur van de tekstmodus te maken. Alleen in de multicolourmodus zijn weer achtergrondkleur van de tekst en van de grafiek identiek op grond van het gelijke register.

Een opmerking nog: het grafiekbeeldscherm is alleen in de programmamodus zichtbaar. Dus als u de HIRES of soortgelijke instructies in direktmodus invoert, werken ze wel, maar wordt er direkt weer naar de tekstmodus teruggeschakeld. Hetzelfde gebeurt natuurlijk na beëindiging van het programma. Wilt u de grafiekmodus in het programma uitschakelen, dan gebruikt u de NRM-instructie of de CSET 0/1-instructie. Door een trucje is het mogelijk ook buiten het programma de grafiek zichtbaar te houden en ermee te werken. Daarvoor wordt u nu een 3-delig programma gegeven, dat dit effect (dat eigenlijk helemaal niet bedoeld is) bewerkstelligt. Daarbij worden twee u nog onbekende instructies gebruikt:

1. MEM, dat in hoofdstuk 13 verklaard wordt en

2. BLOCK, een hierna te verklaren grafiekinstructie

U moet dit eerst maar even aannemen.

```
10 HIRES 6,7 : REM GRAFIEK INSCHAKELEN
20 MEM : REM KARAKTERSET VERSCHUIVEN
30 BLOCK 0,0,319,105,0 : REM BOVENSTE GRAFIEKDEEL UITWISSEN
```

Een kleine beperking moet er gemaakt worden: u kunt geen enkele kleurbepaling uitvoeren, behalve door het effect, dat de nu geschreven tekst als een klein kleurvierkant zichtbaar wordt.

Voorbeeld:

```
100 REM *********************
110 REM **                 **
120 REM **  HIRES-VOORBEELD **
130 REM **                 **
140 REM *********************
150 REM
```

```
160 FOR AK=0 TO 14 : REM ACHTERGRONDKLEUR
170 HIRES 15,AK : REM GRAFIEK TEKENEN ,WISSEN EN KLEUREN INSTELLEN
180 FOR X=59264 TO 59583
190 POKE X, %11110000 : REM BEELDSCHERMOPBOUW:STREPEN
200 NEXT X
210 NEXT AK : REM VOLGENDE ACHTERGRONDKLEUR
220 REM NA EINDIGEN,TERUGKEER NAAR TEKST
```

Dit programma bestaat uit twee in elkaar genestelde FOR...NEXT-lussen. De buitenste van de twee laat de achtergrondkleur keer op keer in totaal 15 maal wisselen, terwijl de binnenste regel lange, verticale strepen door inPOKEn van de desbetreffende waarden in de betrokken geheugencellen tekent (zie inleiding bij dit hoofdstuk). Bij dit inPOKEn wordt geen kleur gedefinieerd. Daaraan kunt u zien, dat de kleur van de afzonderlijke punten (hier grijs 3 --- kleurcode 15) al door de HIRES-instructie gegeven is. Wisselt u eens in plaats van de achtergrondkleur de puntkleur in de buitenste lus.

12.2.2 MULTI

Format	MULTI k1, k2, k3
Parameter	k1: Puntkleur van het kleurregister 1 (0-15)
	k2: Puntkleur van het kleurregister 2 (0-15)
	k3: Puntkleur van het kleurregister 3 (0-15)
	MULTI 4, 5, 6
Functie	Omschakelen binnen de grafiekmodus op multicolourmodus (MC) en bepalen van de 3 puntkleuren

Met multicolour (MC) staan u in totaal 4 kleuren per 8*8- (of 4*8-, omdat de grafiekoplossing in x-richting immers kleiner is) blok ter beschikking. De MULTI-instructie werkt anders dan de HIRES-instructie. Terwijl de laatste een hele serie functies overneemt (inschakelen van de grafiek, inschakelen van het HGR-proces, bepaling van de achtergrond- en de puntkleur), schakelt MULTI alleen van HGR-

op MC-modus over en definieert de kleuren. Dus als de grafiek tijdens het maken zichtbaar moet zijn, moet ze eerst door HIRES (of CSET 2, zie beneden) ingeschakeld worden. In de 3 parameters bij de MULTI-instructie bepaalt u drie van de vier kleurregisters die betrekking hebben op de kleur van een punt. Welk register voor welk punt uitgekozen wordt, wordt immers door de 2-bits-informatie van een punt in het grafiekgeheugen vastgelegd. Het bij deze parameters ontbrekende kleurregister is het kleurregister voor de achtergrond, dat door inPOKEn (POKE 53248 + 33,f0) of door gebruik van de instructie COLOUR (zie hoofdstuk 9) veranderd kan worden. Over de adressen en de verschillende kleurregisters geeft de inleiding u uitvoerig informatie. Dus: k1/2 in het video-RAM / k3 in het kleur-RAM. U weet ook, dat het kleurregister 3 (of beter de kleurregisters 3) die kleur nummer 3 voor elk 8*8 (of 4*8)-blok geeft, in het kleurgeheugen van de tekstmodus ligt. Dit brengt echter ook meteen een paar problemen met zich mee. Als de multicolourmodus ingeschakeld wordt, wordt daardoor ook het kleurregister 3, dus het kleurgeheugen, met de betreffende waarde gevuld. Daardoor zullen alle tekens die op dat moment in het tekstvenster staan, ongeacht of er net grafiek of tekst zichtbaar is, de kleur krijgen die u als k3 in de MULTI-instructie hebt ingevoerd. Probeert u het maar eens. Schrijf het beeldscherm vol met tekens en voer dan in:

MULTI1,2,3

U zult het effect meteen merken. Verandert u de derde parameter in dit voorbeeld. Dan ziet u dat, als u met veel moeite een kleurrijke tekst gemaakt heeft, al dat werk weer door de MULTI-instructie teniet gedaan wordt. Omgekeerd gebeurt er natuurlijk ook iets als u grafiek in de multicolourmodus gemaakt hebt en daarna tekst in het tekstvenster schrijft. Meteen verandert dan op deze plaatsen in de grafiek de kleur 3.

U kunt dit effect natuurlijk ook juist voor uw eigen doel gebruiken. In aansluiting op deze uiteenzettingen zullen we met het volgende voorbeeld enige uitwerkingen van de MULTI-instructie aanschouwelijk maken.

```
100 REM **********************
110 REM **                  **
120 REM **  MULTI-VOORBEELD **
130 REM **                  **
140 REM **********************
150 REM
160 COLOUR 0,6
170 HIRES 7,6 : REM GRAFIEK TEKENEN, WISSEN EN ACHTERGRONDKLEUR =
    BLAUW
180 AD = 60000 : REM BASISADRES UITGANGSPUNT
```

```
190 FOR X=0 TO 7
200 POKE X+AD , %00000000 : REM 1 STREEP MET KLEUR 0 (ACHTERGROND)
    TEKENEN
210 NEXT X : REM VOLGENDE BYTE
220 FOR X=8 TO 15
230 POKE X+AD , %01010101 : REM 1 STREEP MET KLEUR 1 TEKENEN
240 NEXT X : REM VOLGENDE BYTE
250 FOR X=16 TO 23
260 POKE X+AD , %10101010 : REM 1 STREEP MET KLEUR 2 TEKENEN
270 NEXT X : REM VOLGENDE BYTE
280 FOR X=24 TO 31
290 POKE X+AD , %11111111 : REM 1 STREEP MET KLEUR 3 TEKENEN
300 NEXT X : REM VOLGENDE BYTE
310 REM
320 PAUSE 5 : REM NOG STEEDS IS ALLEEN HGR INGESCHAKELD
330 FOR X=0 TO 15
340 MULTI X,7,5 : REM MULTICOLOR INSCHAKELEN EN 3 KLEUREN KIEZEN:
350 REM KLEUR 1: VARIABEL
360 REM KLEUR 2: GEEL
370 REM KLEUR 3: GROEN
380 PAUSE 1
390 NEXT X
400 REM
410 REM VOLGENDE KLEUR LATEN FLITSEN
420 FOR X=0 TO 15
430 MULTI 7,X,5 : REM MULTICOLOR INSCHAKELEN EN 3 KLEUREN KIEZEN
440 REM KLEUR 1: GEEL
450 REM KLEUR 2: VARIABEL
460 REM KLEUR 3: GROEN
470 PAUSE 1
480 NEXT X
490 REM
500 REM VOLGENDE KLEUR LATEN FLITSEN
510 FOR X=0 TO 15
520 MULTI 7,5,X : REM MULTICOLOR INSCHAKELEN EN 3 KLEUREN KIEZEN
530 REM KLEUR 1: GEEL
540 REM KLEUR 2: GROEN
550 REM KLEUR 3: VARIABEL
560 PAUSE 1
570 NEXT X
```

Zoals u ziet, kunnen delen van de grafiek naar kleur worden opgeslagen in het geheugen en van daaruit weer afzonderlijk worden opgeroepen. Steeds worden bepaalde waarden (hier in binaire vorm) in de regels 200, 230, 260 en 290 in het grafiekgeheugen geschreven. We hadden ook in de betreffende kleur met de instructie PLOT kunnen tekenen. Twee bits van iedere waarde bepalen steeds de toewijzing van de punt aan 1 van de 4 kleurregisters. U zult door verandering

van deze regels in het programma deze instructie gemakkelijker begrijpen.

12.2.3 NRM

Format	NRM
Parameter	---
Voorbeeld	NRM
Functie	Uitschakelen van de grafiekmodus en inschakelen van het groot/grafiektekens schrijven

Deze instructie wordt op deze plaats slechts voor de volledigheid vermeld, omdat ze al in paragraaf 9.3.3 uitvoerig uitgelegd werd. Hier is de instructie voor het uitschakelen van de grafiekmodus interessant. Dit wordt in het volgende voorbeeld duidelijk.

```
100 REM ********************
110 REM **                **
120 REM **  NRM-VOORBEELD **
130 REM **                **
140 REM ********************
150 REM
160 PRINT CHR$(14) : REM OP GROOT-/KLEINSCHRIFT OMSCHAKELEN
170 PRINT CHR$(147) : REM TEKST OP BEELDSCHERM WISSEN
180 CENTRE "OP HET MOMENT IS DE KLEINSCHRIFT-" : PRINT
190 CENTRE "MODUS INGESCHAKELD":PRINT
200 PRINT : REM LEGE REGEL
210 CENTRE "KIJK EENS GOED NAAR DE VOLGENDE" : PRINT
220 CENTRE "MELDING!"
230 PAUSE 9
240 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
250 CENTRE "DIT IS HET TEKSTGEDEELTE": REM MELDING VAN TEKSTGEDEELTE
260 REM
270 FOR Y=1 TO 5
280 PAUSE 2
290 HIRES 7,6 : REM HGR AAN + WISSEN / PUNTKLEUR=GEEL / ACHTERGROND=
    BLAUW
```

```
300 FOR X=59264 TO 59583 : REM BESLAAT EEN REGEL
310 POKE X, %11110000 : REM STREPEN TEKENEN
320 NEXT X
330 PAUSE 2
340 NRM: REM TEKSTMODUS INSCHAKELEN (OP HOOFDLETTERS)
350 NEXT Y : REM 5 MAAL WISSELEN
360 PRINT : PRINT : REM 2 LEGE REGELS
370 CENTRE "ZOALS U ZIET, WERD DOOR NRM OP" : PRINT
380 CENTRE "HOOFDLETTERS OVERGESCHAKELD"
390 REM VERANDERT U DE WAARDE VAN %11110000 IN REGEL 220 EENS
400 REM BIJV. IN %11000000 OF 11001100 OF ...
```

12.2.4 CSET 2

Format	CSET 2
Parameter	------
Voorbeeld	CSET 2
Functie	Inschakelen van de grafiekmodus zonder de grafische voorstelling uit te wissen

Met CSET wordt u een instructie aan de hand gedaan, die een serie verschillende en op het eerste gezicht onsamenhangende functies uitvoert. Haar verschillende verschijningsvormen zullen later nog een keer in het middelpunt van onze belangstelling staan; namelijk wanneer we met de verschillende karaktersets van de CBM 64 te maken zullen hebben (hoofdstuk 13). Daar duikt ze in de vorm van CSET 0 en CSET 1 op. Deze beide instructievarianten schakelen ook onder andere de zichtbare grafiekmodus uit (zoals NRM), maar daarover meer in dat hoofdstuk. Hier en nu interesseert ons haar verschijningsvorm als CSET 2, die ons een andere mogelijkheid van de grafiekmanipulatie in handen geeft: CSET 2 schakelt namelijk - zoals ons al bekend is van de HIRES-instructie - het grafiekbeeldscherm in. Hier echter laat ze een duidelijk verschil zien in vergelijking met HIRES: CSET 2 schakelt in, zonder echter het grafiekbeeldscherm uit te wissen! Als u een grafiek getekend hebt, kunt u tussendoor een keer met NRM of CSET 0/1 (zie hoofdstuk 13) in de tekstmodus omschakelen, om toelichtingen of iets dergelijks te geven en u hebt dan de mogelijkheid het oude plaatje meteen weer zichtbaar te maken. Ook de kleuren blijven behouden. U heeft zelfs de mogelijkheid een grafiek te tekenen,

terwijl u een tekst opgeeft. Later schakelt u dan, als het u uitkomt, met CSET 2 dat plaatje in voor weergave op het scherm. Welke mogelijkheden daarmee voor u ontstaan, hoef ik u niet eerst te zeggen. Een van de vele mogelijke gebruiksvormen wordt u aan de hand van een voorbeeld getoond.

```
100 REM ***********************
110 REM **                   **
120 REM **  CSET 2-VOORBEELD **
130 REM **                   **
140 REM ***********************
150 REM
160 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
170 CENTRE "DIT IS HET TEKSTGEDEELTE": REM MELDING TEKSTGEDEELTE
180 PAUSE 2
190 HIRES 7,2 : REM HGR AAN + WISSEN / PUNTKLEUR = GEEL / ACHTERGROND
    = ROOD
200 REM
210 FOR X=59264 TO 59583 : REM BESLAAT EEN REGEL
220 POKE X, %11110000 : REM STREPEN TEKENEN
230 NEXT X
240 FOR X=1 TO 5
250 PAUSE 2
260 NRM : REM TEKSTMODUS INSCHAKELEN (OP HOOFDLETTERS)
270 PAUSE 2
280 CSET 2 : REM GRAFIEKMODUS INSCHAKELEN ZONDER WISSEN EN KLEUREN
    INSTELLEN
290 NEXT X
```

Zoals u ziet, is er een gelijktijdig proces van grafiek en tekst mogelijk. Zoals altijd kunt u natuurlijk aan dit programma knutselen, als u dat leuk vindt. Speciale aandacht moet u misschien even aan het binaire getal in regel 220 schenken. Die kunt u eventueel (zoals ook in de vorige voorbeelden) veranderen.

12.2.5 LOW COL

Format LOW COL k1,k2,k3

Parameter -k1 : HGR: puntkleur bij het tekenen

MC: Kleurreservering van het kleurregister 1 (0-15) bij het tekenen

-k2 : HGR: geen functie

MC: Kleurreservering van het kleurregister 2 (0-15) bij het tekenen

-k3 : HGR: geen functie

MC: Kleurreservering van het kleurregister 3 (0-15) bij het tekenen

Voorbeeld LOW COL 3,6,2

Functie Bepalen van de puntkleur(en)

Natuurlijk hebt u zich bij lezing van de instructies tot nu toe afgevraagd, of men alle punten slechts in een (HGR) of drie (MC) kleuren kan afbeelden, omdat u deze tot nu toe immers alleen door de instructies HIRES of MULTI een keer aan het begin en dan slechts voor de totale grafiek kon plaatsen. Wel, daar is een oplossing voor. Er is een instructie, die voor alle volgende gezette punten een nieuwe kleur vrijhoudt: LOW COL. Met de daarop volgende parameters bepaalt u de nieuwe kleurencombinatie. In HGR zijn van de drie noodzakelijke waarden k1, k2 en k3 enkel en alleen de twee eerste nodig, omdat we hier immers slechts een puntkleur per 8*8-blok kunnen vastleggen. De tweede waarde dient om de achtergrondkleur van het 8*8-blok vast te leggen. Dit effect wordt in het tweede onder HI COL genoemde voorbeeld geïllustreerd. U moet alle drie de parameters invoeren om geen SYNTAX ERROR te krijgen. De overbodige k3 wordt daarbij genegeerd en kan daarom willekeurig gekozen worden. In multicolour daarentegen, bepalen de beide laatste kleurcodes de twee andere puntkleuren (de achtergrondkleur wordt voor het hele beeld door COLOUR bepaald), die in een 8*8- of 4*8-blokje kunnen voorkomen. Maar het hanteren van deze instructie is niet zo eenvoudig als ze op het eerste gezicht lijkt. Er zijn namelijk verschillen bij de tekening van een punt of een figuur voordat LOW COL wordt gegeven en daarna. Na de inschakeling van de grafiek door de HIRES-instructie (en van de eventuele omschakeling door MULTI op MC) liggen de puntkleuren voor het totale grafiekbeeld al vast en zijn in de aparte geheugencellen van de kleurregisters opgeslagen. Daarom hoeft de kleur bij het tekenen van een punt niet extra geplaatst te worden, maar wordt er gewoon een punt in het grafiekgeheugen gezet. Deze krijgt immers automatisch de kleur die al in het kleurgeheugen staat. Heeft u de LOW COL-instructie ingevoerd, dan moet de kleur van een punt opnieuw en overeenkomstig uw kleurinstructie in de betreffende geheugencellen van de verschillende kleurregisters worden opgeslagen. Omdat nu echter een geheugencel van een kleurregister telkens voor

8*8 (HGR) of 4*8 (MC) punten verantwoordelijk is, krijgen alle andere punten die toevallig al in het blok geplaatst waren dezelfde kleur als het nieuwe onder LOW COL gezette punt, ook wanneer ze oorspronkelijk een andere kleur hadden. Als u later zonder LOW COL, dus zonder gelijktijdige kleurbepaling in hetzelfde 8*8- of 4*8-blok tekent, krijgt het gezette punt de kleur die de reeds gezette punten op deze plaats al hebben. De kleur wordt dus overgenomen. Deze effecten worden met het voorbeeld onder HI COL gedemonstreerd. Let u daar speciaal op de details.

Als u nu bijvoorbeeld in HGR twee figuren (punten, lijnen, cirkels,...) van verschillende kleur wilt tekenen, die waarschijnlijk zeer dicht bij elkaar komen of elkaar raken, moet u erop letten, bij welk figuur u het belangrijker vindt, dat het volledig die kleur behoudt, die eraan was toegekend. Het kan voorkomen, dat delen van deze figuren elkaar overlappen. Nu zijn er twee wegen om dit probleem op te lossen. Of u tekent die figuur, waarvan u de kleur niet zo belangrijk vindt eerst, of u tekent deze zonder kleurbepaling, dus zonder LOW COL, terwijl u de andere met LOW COL construeert. In het eerste geval moet u de volgorde van het tekenen strikt aanhouden, in het tweede geval maakt dat niet uit.

U ziet dus, dat de veelzijdigheid van uw CBM 64 een zeer gecompliceerde grafiekbehandeling met zich meebrengt. De vele mogelijkheden in aanmerking nemend is dat niet verwonderlijk. Als u eenmaal het systeem onder de knie heeft, is de programmering niet moeilijk.

Een voorbeeld:

```
100 REM ***********************
110 REM **                   **
120 REM **  LOW COL-VOORBEELD **
130 REM **                   **
140 REM ***********************
150 REM
160 COLOUR 0,1 : REM VENSTERKLEUR = ZWART
170 AF = 8 : REM AFSTAND TUSSEN DE LIJNEN
180 REM
190 FOR Z=1 TO 2 : REM TWEE OMLOOPEN
200 HIRES 7,0 : REM HGR AAN + WISSEN/ PUNTKLEUR=GEEL / ACHTERGROND=
    ZWART
210 FOR Y=1 TO 15
220 FOR X=0 TO 320
230 PLOT X,Y*AF,1 : REM EEN RIJ PUNTEN TEKENEN
240 NEXT X : REM 320 PUNTEN TEKENEN
250 LOW COL Y,0,0:REM TWEEDE KLEUR INSTELLEN PARAMETER IN HGR
    ONBELANGRIJK
```

```
260 NEXT Y : REM VERANDERING VAN KLEUR
270 PAUSE 5
280 NRM : REM OVERSCHAKELEN OP TEKSTMODUS
290 PRINT CHR$(147):REM BEELDSCHERM WISSEN
300 CENTRE "KIJKT U EENS WAT ER GEBEURD," : PRINT
310 CENTRE "ALS DE LIJNEN VLAK NAAST ELKAAR": PRINT
320 CENTRE "LIGGEN," : PRINT
330 PRINT : REM LEGE REGEL
340 CENTRE "DAN WORDEN ZE UITGEWIST DOOR DE LAGE " : PRINT
350 CENTRE "KLEUROPLOSSING!"
360 PAUSE 9
370 AF = 2 : REM AFSTAND TUSSEN DE LIJNEN
380 NEXT Z : REM NIEUWE GRAFIEK TEKENEN MET MINDER AFSTAND TUSSEN
    DE LIJNEN
```

In het bovenstaande programma duikt in regel 230 een tot nu toe onbekende instructie op: PLOT. PLOT dient voor het tekenen van een punt met de coördinaten x,y en zal later in hoofdstuk 12.3.1 nader besproken worden. Dit voorbeeld demonstreert de kleurbepaling in HGR. Bij het tweede tekenen van de horizontale lijnen door de buitenste FOR...NEXT-lus worden ze zo dicht bij elkaar getekend, dat er overlappingen met de daarvoor getekende lijnen zijn.

12.2.6 HI COL

Format	HI COL
Parameter	-----
Voorbeeld	HI COL
Functie	Opheffen van de LOW COL-modus = tekenen zonder kleurbepaling

Voor het lezen van deze instructie moet u de werking van de LOW COL-instructie doorgewerkt en begrepen hebben, omdat dit in het volgende verondersteld wordt. Gewoonlijk wordt na het initialiseren van de grafiek-modus door HIRES elke geplaatste punt zonder aparte kleurbepaling getekend. Pas na de invoer van LOW COL wordt met elke geplaatste punt de aangegeven kleur in het video-RAM of bij multicolour bovendien nog in het kleurram bepaald. Deze

kleurbehandeling bracht, zoals we onder LOW COL gezien hebben, enige problemen met zich mee. Vanwege de beperkte beschikbaarheid van kleurmogelijkheden bij de Commodore 64 hadden we al in paragraaf 12.2.5 de noodzaak onderkend van een instructie die de werking van een kleurbepaling weer teniet doet, zodat net als bij het begin van de grafiekactiviteiten figuren getekend kunnen worden, zonder dat meteen de kleur in het kleurgeheugen geplaatst wordt. Deze functie heeft de HI COL-instructie zonder parameters. Welke relevantie deze instructie in het kader van de kleurbehandeling heeft, zal u door het volgende voorbeeld van een combinatie van HI COL en LOW COL, een onvoorwaardelijk samenhorig paar, verduidelijkt worden.

```
100 REM ***********************
110 REM **                   **
120 REM **  HI COL-VOORBEELD **
130 REM **                   **
140 REM ***********************
150 REM
160 COLOUR 0,1 : REM VENSTERKLEUR = ZWART
170 REM
180 HIRES 6,0 : REM HGR AAN + WISSEN / PUNTKLEUR =BLAUW / ACHTERGROND
    =ZWART
190 FOR Y=0 TO 11
200 FOR X=0 TO 319
210 PLOT X,Y*16,1 : REM EEN RIJ PUNTEN TEKENEN
220 NEXT X : REM 320 PUNTEN (EEN REGEL) TEKENEN
230 LOW COLY+1,0,0 : REM 2. KLEUR PLAATSEN
235 REM :TWEE ANDERE PARAMETERS ZIJN HIER NIET VAN BELANG
240 FOR X=0 TO 319
250 PLOT X,Y*16+8,1 : REM EEN RIJ PUNTEN TEKENEN
260 NEXT X : REM 320 PUNTEN (EEN REGEL) TEKENEN
270 HI COL : REM MET OUDE KLEUR = ZONDER KLEUR TEKENEN (BLAUW)
280 NEXT Y : REM VERANDERING VAN KLEUR
290 PAUSE 5
300 NRM : REM TEKSTMODUS INSCHAKELEN
310 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
320 CENTRE "LET U ER EENS OP HOE BIJ" : PRINT
330 CENTRE "LOW COL MET KLEURINSTELLING EN BIJ" : PRINT
340 CENTRE "HI COL ZONDER KLEURINSTELLING" : PRINT
350 CENTRE "WORDT GETEKEND"
360 PAUSE 10
370 CSET 2 : REM GRAFIEK INSCHAKELEN ZONDER WISSEN
380 FOR Y=0 TO 9
390 FOR X=0 TO 199
400 PLOT Y*32,X,1 : PLOT Y*32+1,X,1 : REM BREEDTE VERDUBBELEN
410 REM EEN KOLOM PUNTEN TEKENEN
420 NEXT X : REM 200 PUNTEN (EEN KOLOM) TEKENEN
```

```
430 LOW COL Y,0,0
440 FOR X=0 TO 199
450 PLOT Y*32+16,X,1 : PLOT Y*32+17,X,1 : REM BREEDTE VERDUBBELEN
460 REM EEN KOLOM PUNTEN TEKENEN
470 NEXT X : REM 200 PUNTEN
480 HI COL : REM MET OUDE KLEUR = ZONDER KLEUR TEKENEN (BLAUW)
490 NEXT Y : REM VERANDERING VAN KLEUR
500 PAUSE 20
```

Zoals u misschien gemerkt zult hebben, leunt dit voorbeeld tegen het programma voor de instructie LOW COL aan. Eerst een kleine programmabeschrijving.

In dit voorbeeld van de combinatie van LOW COL en HI COL worden aan het begin horizontale lijnen met een constante afstand van 8 punten afwisselend met en zonder kleurbepaling getekend. Daarbij wordt de algemene (in HIRES aangegeven) puntkleur blauw gehouden. De met LOW COL geplaatste lijnen daarentegen wisselen steeds in hun kleuren. Op zich is dit niet nieuw meer, omdat u dit al van het vorige voorbeeld bekend is. Er komt nu echter een tweede element bij, dat u de samenhang, of de verschillen tussen punten met en zonder kleurbepaling duidelijk maakt. Er worden (vanaf regel 370) verticale lijnen met een afstand van 16 punten eveneens telkens met en zonder kleurbepaling getrokken. Daarbij moet u opvallen, hoe de kleurbepaling van de onder LOW COL getekende horizontale lijnen de kleur van de omliggende punten beïnvloedt, terwijl in tegenstelling hiermee de onder HI COL geplaatste lijnen door de verschillende kleuren van de reeds geplaatste punten (lijnen) beïnvloed worden. De verticale lijnen worden telkens dubbel zo dik getekend, om de kleur van de aparte lijnen te kunnen herkennen.

Het volgende voorbeeld, eigenlijk een voorbeeld voor LOW COL, wordt op grond van de opduikende HI COL-instructie pas hier gegeven. Het belicht een ander aspect van de LOW COL-instructie, namelijk dat van het plaatsen van de achtergrondkleur na gebruik van LOW COL.

```
100 REM **************************
110 REM **                      **
120 REM **  LOW COL-VOORBEELD 2 **
130 REM **                      **
140 REM **************************
150 REM
160 HIRES 5,6 : REM GRAFIEK AAN/ACHTERGROND = BLAUW/PUNTEN = GROEN
170 LOW COL 1,5,0 : REM ACHTERGROND = GROEN/PUNTEN = WIT
180 REM
190 REM EEN HORIZONTALE LIJN TEKENEN (MET KLEUR):
200 FOR X=0 TO 319
```

```
210 PLOT X,66,1 : REM PUNT TEKENEN
220 NEXT X
230 PAUSE 2
240 LOW COL 0,1,0 : REM ACHTERGROND = WIT/PUNTEN = ZWA
250 REM
260 REM DE HORIZONTALE LIJN MET KLEUR WISSEN
270 FOR X=0 TO 319
280 PLOT X,133,0 : REM PUNT WISSEN
290 NEXT X
300 PAUSE 2
310 HI COL: REM OVERGAAN OP TEKENEN ZONDER KLEUR
320 REM
330 REM VERTICALE LIJN TEKENEN (ZONDER KLEUR)
340 FOR Y=0 TO 199
350 PLOT 160,Y,1 : REM PUNT TEKENEN
360 NEXT Y
370 WAIT198,255 : REM WACHTEN OP TOETSAANSLAG
```

Naast de puntkleur wordt eveneens de achtergrondkleur onder LOW COL geplaatst. Dat is mogelijk, omdat de achtergrondkleur net als de puntkleur uit het video-RAM komt en daarom per 8*8-blok verschillend kan zijn. Het tweede vraagstuk dat door dit voorbeeld aanschouwelijk gemaakt wordt, is het plaatsen van de kleur, onafhankelijk, of het aangesproken punt geplaatst of uitgewist wordt, wat door de tekenmodus (zie PLOT) aangegeven wordt. Hoewel we dus een punt uitwissen, wordt de kleur in de kleurregisters veranderd. Moet de oorspronkelijke kleur worden teruggeplaatst, dan gebruikt u die waarde met de LOW COL-instructie, die overeenkomt met uw normale HI COL-kleur (dus de kleur, die in de HIRES-instructie geplaatst werd).

12.2.7 Testopdrachten

Deze keer wordt de test iets moeilijker, omdat we met een zeer gecompliceerd thema te maken hebben. Lukt het niet meteen, leest u dan de betreffende passages nog een keer goed door.

1. Hoe initialiseert u bij het begin van uw grafiekwerk de multicolourgrafiek? Door:

   A) HIRES ...

B) HIRES ... : MULTI ...
C) CSET 2 : MULTI ...
D) CSET 2

2. Wat doet de instructie NRM ?

A) slechts het uitschakelen van de grafiek
B) uitschakelen van de grafiek-modus en instellen van de groot/klein-modus
C) uitschakelen van de grafiek-modus en instellen van de hoofdletters/grafiek-modus
D) terugplaatsen van de kleur naar de beginwaarden

3. U tekent een punt, nadat u LOW COL ingevoerd heeft. Wat is er te zien?

A) er vindt slechts een wisseling van de kleuren plaats
B) alle punten in hetzelfde 8*8-veld krijgen dezelfde kleur
C) de achtergrondkleur voor dit punt wordt geplaatst
D) het kleurnummer van de puntkleur wordt met 1 verlaagd

12.3 Instructies voor de grafiek

Al het droge werk in de voorafgaande paragrafen zal nu door de egenlijke grafiekinstructies beloond worden. Nu kunt u grafiek maken met het tekenen van punten, strepen, cirkels, velden en nog veel meer. Kijkt u en probeert u, want nu begint het niemandsland. U heeft de mogelijkheid dingen te maken die iemand anders voor u nog nooit op het scherm gebracht heeft. Of u gebruikt de grafiek alleen om spelen te maken, waarbij de sprite-instructies zeer nuttig zullen blijken. Kortom: nu komen we tot het kernthema van onze grafiekparagraaf: de tekeninstructies. Bij elk van de volgende grafiekinstructies kunt u naast de andere parameters die van geval tot geval verschillend zijn en ter plaatse verklaard worden, de u al bekende tekenmodus kiezen. Daarmee kunt u naar eigen goeddunken bepalen, of u de aangesproken figuur wilt tekenen, uitwissen of inverteren. Daarmee staat u per instructie een hele serie mogelijkheden van figuurpresentatie open, omdat elke instructie natuurlijk eveneens in multicolour functioneert. Onder x- of

y-coördinaten wordt natuurlijk die positionering verstaan, die Simon's BASIC zoals onder 12.2 beschreven uitvoert.

Nog een urgente tip zou ik u mee willen geven: probeert u zoveel mogelijk alle voorbeelden uit, u zult er geen spijt van krijgen! Slechts zo leert u de fascinerende mogelijkheden van de aparte instructies kennen. Belangrijk is, dat u probeert zoveel mogelijk in de opgegeven programma's te veranderen, om de daaruit resulterende problemen te leren kennen.

12.3.1 PLOT

Format	PLOT x,y,zm
Parameter	x : x-coördinaten van het punt MC: 0-159 / HGR: 0-139
	y : y-coördinaten van het punt MC: 0-199 / HGR: 0-199
	tm: tekenmodus
Voorbeeld	PLOT 160, 100, 1
Functie	Plaatsen van een punt bij x,y

Met PLOT, de eenvoudigste tekeninstructie, wordt u in het rijk der grafiek ingevoerd. PLOT plaatst alleen maar een enkel punt op het beeldscherm, waarvan u de coördinaten met x en y aangeeft. Let u op, dat het nulpunt van het imaginaire coördinatensysteem in de bovenste linkerhoek van het beeldscherm ligt. Met tm geeft u de tekenmodus aan. Dat wil zeggen: bij z=0 wordt een eventueel geplaatst punt op de aangegeven plaats uitgewist, bij z=1 wordt hij geplaatst, en bij z=2 verschijnt op het genoemde coördinaat een punt, als er daarvoor nog niet een stond en omgekeerd wordt er een punt uitgewist, als er reeds een punt geplaatst was. Met behulp van deze instructie kunt u de vaardigheid met de twee coördinaten en de tekenmodus nog zeer gemakkelijk oefenen en de verschillende effecten uitproberen. In aansluiting hierop zullen u twee voorbeelden (in een programma) gedemonstreerd worden, die u de werkwijze van de PLOT-instructie laten zien.

Voorbeeld:

```
100 REM ********************
110 REM **                **
120 REM **  PLOT-VOORBEELD **
130 REM **                **
140 REM ********************
160 REM
170 HIRES 8,9 : REM GRAFIEK AAN/PUNTKLEUR = ORANJE/ACHTERGROND =
    BRUIN
180 FOR X=1 TO 319 STEP .5
190 PLOT X,60 * SIN(X/25) + 100 ,1
200 NEXT X
210 PAUSE 5
220 HIRES 8,9 : REM GRAFIEK WISSEN
230 FOR Y=1 TO 0 STEP -1 : REM EERSTE TEKENEN / TWEEDE WISSEN
240 FOR X=103 TO 290 STEP .5
250 PLOT 40 * SIN(X/30) + 160 , 50 * SIN(X/25) + 100 ,Y
260 NEXT X
270 PAUSE 1
280 NEXT Y : REM NOGMAALS WISSEN
290 WAIT 198,255: REM WACHTEN OP TOETSAANSLAG
```

In het eerste gedeelte van dit voorbeeld wordt u aan de hand van een eenvoudige sinuscurve de functie van de PLOT-instructie verklaard. Verandert u rustig een keer de parameters. In het tweede gedeelte van het programma (vanaf regel 220) wordt u met een interessante toepassing geconfronteerd, die gelijktijdig de functie van de tekenmodus beschrijft (hier in het geheugen y opgeslagen). Regel 290 werd reeds op een andere plaats beschreven en heeft als inhoud de instructie, die op een toetsaanslag wacht. Hij zal hierna vaker gebruikt worden, omdat de computer na het beëindigen van het programma steeds naar de tekstmodus terugkeert en u daarmee het verdere zicht op de grafiek belemmert.

12.3.2 TEST

Format TEST (x,y)

Parameter x : x-coördinaten van het punt MC: 0-159 / HGR:

	0-319
	y : y-coördinaten van het punt MC: 0-199 / MGR: 0-199
Voorbeeld	TEST (100,150)
Functie	Testen van de aangegeven beeldschermcoördinaten op een geplaatst punt

Stel, u schrijft een programma, dat welke punten dan ook, hetzij door de zojuist beschreven PLOT-instructie hetzij door andere grafiekinstructies, op het beeldscherm brengt. In de loop van het programma echter wilt u weten, of er op een bepaalde plaats op het grafiekbeeldscherm een keer een punt geplaatst werd. Dat kan bijvoorbeeld gebeuren om snijpunten, botsingen (bij de uitvoering van spelen) of begrensde beeldschermgebieden vast te stellen. Wel, die taak neemt de instructie TEST op zich. Hierbij bepalen de als argument tussen haakjes geplaatste parameters x en y op de bekende manier de coördinaten van het te testen punt.

TEST wordt automatisch als een rekenkundige functie behandeld. Rekenkundige functies zijn bijvoorbeeld SIN, COS, SQR of de Simon's Basic-instructie LIN. Met behulp van deze functies wordt een waarde berekend, die bij verdere berekeningen gebruikt of direkt in een geheugencel opgeslagen kan worden:

```
A = TEST (0,0)
B = 3*5 + TEST (0,10)
```

De instructie TEST levert slechts een van de twee mogelijke waarden: 0 of 1. Welke waarde uiteindelijk door het resultaat geleverd wordt, hangt ervan af of op de aangesproken plaats een punt geplaatst is of niet. Is op de plaats met de coördinaten x,y een punt afwezig, dan neemt de functie de waarde 0 aan. In het omgekeerde geval echter, dus als er een punt aanwezig is, verschijnt de waarde 1. Met deze waarden wordt dan, wanneer dat gewenst is, in de volgende berekeningen verder gerekend. Met deze instructie kunnen vooral aardige spelletjes gemaakt worden. Daarom werd in aansluiting op het voorbeeld, dat de functie van deze instructie verklaart, een klein maar zeer amusant spelletje toegevoegd, dat enkel en alleen de voorhanden-zijnde grafiekinstructies uitbuit.

Voorbeelden:

```
100 REM **********************
110 REM **                  **
120 REM **  TEST-VOORBEELD 1 **
130 REM **                  **
```

```
140 REM ***********************
150 REM
160 HIRES 6,7 : REM GRAFIEK AAN + WISSEN
170 T1 = TEST(100,100) : REM TESTEN VOOR HET PLAATSEN
180 PAUSE 1
190 PLOT 100,100,1 : REM PUNT PLAATSEN
200 T2 = TEST(100,100) : REM TESTEN NA HET PLAATSEN
210 PAUSE 1
220 NRM : REM GRAFIEK UIT
230 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
240 PRINT "TEST VOOR HET PLAATSEN:" T1
250 PRINT "TEST NA   HET PLAATSEN:" T2
```

```
100 REM **********************
110 REM **                  **
120 REM ** TEST-VOORBEELD-2 **
130 REM **                  **
140 REM **********************
150 REM
160 REM   ******************
170 REM   **              **
180 REM   **  SUPER-SNAKE **
190 REM   **              **
200 REM   ******************
210 REM
220 COLOUR 10,1 : REM VENSTERKLEUR LICHTROOD
230 HIRES 1,2 : REM GRAFIEK AAN/PUNTKLEUR = WIT/ACHTERGROND = ROOD
240 G = 4 : REM SNELHEID***********************************
250 X = -G : Y = 0 : REM STARTRICHTING: LINKS
260 A = 160 : B = 100 : REM STARTCOORDINATEN
270 GET A$ : REM TOETS INLEZEN
280 IF A$ = "I" THEN X =  0 : Y = -G : REM OMHOOG
290 IF A$ = "M" THEN X =  0 : Y =  G : REM NAAR BENEDEN
300 IF A$ = "J" THEN X = -G : Y =  0 : REM LINKS
310 IF A$ = "K" THEN X =  G : Y =  0 : REM RECHTS
320 A = A+X : B = B+Y : REM DOORGAAN
330 IF A<0 OR A>319 OR B<0 OR B>199 THEN GOTO 370
340 IF TEST (A,B) = 1 THEN GOTO 370
350 PLOT A,B,1 : REM PUNT PLAATSEN
360 GOTO 270
370 REM
380 REM **********************
390 REM **                  **
400 REM **   BOTS ROUTINE   **
410 REM **                  **
420 REM **********************
430 REM
440 FOR C=1 TO 15
```

```
450 FOR D=0 TO 15
460 COLOUR D,1 : REM VENSTER FLITSEN
470 NEXT D
480 NEXT C
490 A = INT( RND(1) * 319/G ) * G
500 B = INT( RND(1) * 199/G ) * G
510 COLOUR 10,1
520 Z = Z+1 : REM TELLER
530 IF Z = 20 THEN GOTO 550
540 GOTO 270
550 REM
560 REM ************
570 REM **        **
580 REM **  EINDE **
590 REM **        **
600 REM ************
610 REM
620 NRM : REM GRAFIEK UIT
630 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
640 PRINT AT(0,10) "" : REM CURSOR POSITIONEREN
650 CENTRE "GEEN PANIEK!!!"
660 POKE 198,0 : REM TOETSEN INVOER WISSEN
670 WAIT 198,255 : REM WACHTEN OP TOETSAANSLAG
```

Het spel functioneert als volgt. Op het grafiekbeeldscherm beweegt ononderbroken een steeds langer wordende slang. U moet met de toetsen

I voor omhoog
M voor omlaag
J voor naar links
K voor naar rechts

overeenkomend met de toetsenvolgorde:

```
      I
  J       K
      M
```

op het toetsenbord de bewegingsrichting van de slang zo sturen, dat ze nooit uit het beeldschermvenster gaat, noch tegen een gedeelte van zichzelf aanstoot. In beide gevallen krijgt u een vermindering van punten. Daarna verschijnt de slang weer ergens op het beeldscherm. Bij 20 van zulke terugslagen is het spel afgelopen. U heeft natuurlijk de mogelijkheid, het spel naar uw eigen idee bij te vijlen en te veranderen. U kunt het voor twee tegen elkaar spelende spelers

herschrijven, punten tellen, in plaats van de besturing per toetsenbord de joystick nemen enzovoort. Het spel is speciaal zo gemaakt dat u veranderingen goed en gemakkelijk kunt uitvoeren. De snelheid van de slang kunt u in regel 240 gemakkelijk veranderen (G bevat de snelheid = pasafmeting per ronde). In de regels 280-310 worden de vier toetsen getest op een eventuele aanslag en dienovereenkomstig wordt de richting gestuurd. Dan berekent regel 320 de nieuwe coördinaten. De parameters x en y bepalen telkens het aantal passen per ronde in x- of y-richting. A en B hebben als inhoud de actuele coördinaten van het punt, die in regel 330 daarop getoetst worden, of ze nog binnen het grafiekvenster liggen.

Het interessantst voor ons bij dit voorbeeld is echter de invoer van de TEST-instructie in regel 340. Hier wordt telkens de actuele punt getoetst. Is er geplaatst (TEST=1), dan wordt er een gedeelte van de slang geraakt. Verder (TEST=0) wordt er nu op deze plaats een nieuwe punt getekend. Misschien inspireert dit voorbeeld u tot prestaties, die bij uw familieleden of vrienden veel weerklank zullen vinden.

12.3.3 LINE

Format	LINE x1, y1, x2, y2, tm
Parameter	x1 : x-coördinaat van het beginpunt van de lijn MC: 0-159 / HGR: 0-319
	y1 : y-coördinaat van het beginpunt van de lijn MC: 0-199 / HGR: 0-199
	x2 : x-coördinaat van het eindpunt van de lijn MC: 0-159 / HGR: 0-319
	y2 : y-coördinaat van het eindpunt van de lijn MC: 0-199 / HGR: 0-199
	tm : tekenmodus
Voorbeeld	LINE 0,0,310,199,1
Functie	Tekenen van een lijn

Met de PLOT-instructie zou u eigenlijk alle figuren kunnen tekenen die denkbaar zijn. Maar stelt u zich eens voor, hoeveel moeite en tijd het in beslag zou nemen, als u bijvoorbeeld alleen met de PLOT-instructie een lijn zou willen tekenen. U zou elk afzonderlijk punt van de lijn (na de 2-puntenvergelijking) in BASIC moeten berekenen en daarna plaatsen. Hoe lang dat duurt, kunt u zich voorstellen! Om deze hindernis uit de weg te ruimen, werd in Simon's BASIC een hele serie instructies ontwikkeld, die figuren tekenen, die direkt klaar zijn. Een daarvan is de zogenaamde LINE-instructie. Het maakt u op een eenvoudige manier mogelijk een lijn te tekenen, waarvan de twee grenspunten door u kunnen worden aangegeven. U hoeft u verder nergens zorgen over te maken, de rest wordt door Simon's BASIC afgehandeld. U zult wel merken, dat alle instructies om figuren te maken in Simon's BASIC niet erg snel zijn, maar daarvoor zijn er uiteindelijk speciale grafiekprogramma's, die wat dit betreft de veeleisendste personen tevreden stellen. Zo kunt u gemakkelijker en sneller over de verschillende functies beschikken, die in het middelpunt van de grafiekpresentatie staan. De eerste twee parameters (x1 en y1) stellen bij LINE de coördinaten voor, waaruit een lijn getrokken moet worden (=1e punt), gevolgd door x2 en y2, die bovendien het eindpunt van de lijn vastleggen (=2e punt). Daarbij is het niet altijd onbelangrijk, welk punt begin- of eindpunt is. Als u bijvoorbeeld van de coördinaten 100,110 een lijn naar 200,150 getekend hebt en u wilt deze uitwissen, dan moet u deze lijn in dezelfde richting uitwissen, als waarin u haar getekend hebt. Verder wordt uw lijn alleen puntsgewijs uitgewist (slechts delen van de lijn worden uitgewist), wat waarschijnlijk niet in uw belang is. Dit geldt namelijk niet eens voor alle lijnen, omdat de mate van het uitwissen van de hoek afhangt, die de lijn met de horizontale lijn maakt. Dus:

```
10 HIRES 2,3
20 LINE 100,110,200,150,1
30 PAUSE 1
40 LINE 200,150,100,110,0:REM FOUT!!!
50 WAIT 198,255
```

maar:

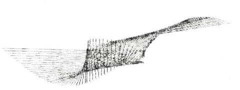
```
10 HIRES 2,3
20 LINE 100,110,200,150,1
30 PAUSE 1
40 LINE 100,110,200,150,0:REM GOED!!!
50 WAIT 198,255
```

Iets dergelijks geldt natuurlijk voor het inverteren. Met LINE zijn er heel mooie effecten te produceren, waar in de volgende voorbeelden een paar hints voor worden gegeven.

```
100 REM *********************
110 REM **                 **
120 REM **  LINE-VOORBEELD **
130 REM **                 **
140 REM *********************
150 REM
160 REM    ****************
170 REM    **            **
180 REM    **  FIGUUR 1  **
190 REM    **            #*
200 REM    ****************
210 REM
220 HIRES 6,7
230 FOR X=1 TO 319 STEP 4
240 LINE 0,0,X,40 * SIN(X/20) + 100,1
250 NEXT X
260 PAUSE 10
270 HIRES 6,7
280 REM
290 REM    ****************
300 REM    **            **
310 REM    **  FIGUUR 2  **
320 REM    **            **
330 REM    ****************
340 REM
350 FOR X=1 TO 350 STEP 1
360 A = 50 * SIN(X/10) + 160
370 B = 60 * SIN(X/20) + 100
380 C = 70 * SIN(X/30) + 160
390 D = 80 * SIN(X/40) + 100
400 LINE A,B,C,D,1
410 NEXT X
420 PAUSE 10
430 HIRES 6,7
440 REM
450 REM    ****************
460 REM    **            **
470 REM    **  FIGUUR 3  **
480 REM    **            **
490 REM    ****************
500 REM
510 FOR X=1 TO 319 STEP 4
520 A = X
530 B = 60 * SIN(X/60) + 100
```

```
540 C = 40 * SIN(X/25) + 160
550 D = 20 * SIN(X/30) + 100
560 LINE A,B,C,D,1
570 NEXT X
580 WAIT 198,255 : REM WACHTEN OP TOETSAANSLAG
```

Zoals u ziet, laten alle parameters zich niet alleen eenvoudig benoemen, maar kunnen ze ook op meer of minder gecompliceerde wijze berekend worden. U kunt daarin naar hartelust veranderingen aanbrengen. Al doende komt u zeker (vooral, als u wat meer van wiskunde weet) achter een paar foefjes om figuren doelgericht te veranderen of te ontwerpen. Zo werd in het eerste voorbeeld elk punt van een eenvoudige sinuscurve met het punt 0,0 verbonden.



12.3.4 REC

Format REC x, y, b, h, tm

Parameter x : x-coördinaat van de linker bovenhoek van de rechthoek

MC: 0-159 / HGR: 0-319

y : y-coördinaat van de linker bovenhoek van de rechthoek

MC: 0-199 / HGR: 0-199

b : breedte van de rechthoek

MC: 0-159 / HGR: 0-319

h : hoogte van de rechthoek

MC: 0-199 / HGR: 0-199

tm: tekstmodus

Voorbeeld	REC 20,40,100,50,1
Functie	Tekenen van een rechthoek

Een andere kant-en-klare instructie om figuren te maken, is de REC-instructie. Met behulp van REC is het mogelijk om een rechthoek van variabele vorm, grootte en positie op het grafiekbeeldscherm te brengen. U heeft daarmee een middel op kaders om bepaalde figuren te trekken, rechthoeken vol effecten of eenvoudigweg rechthoeken te creëren. De positie van de gewenste vierhoek geeft u door de coördinaten van de linker bovenhoek van uw figuur aan. Om nu de grootte van de rechthoek en daarmee zijn uitbreiding naar beneden en rechts te bepalen, plakken we aan de eerste twee parameters van de REC-instructie twee andere vast, die aan de computer de breedte (b) en de hoogte (h) van onze figuur geven. Daarbij worden telkens alle randen van de rechthoek meegeteld. Uiteraard kunnen deze waarden dezelfde grootte als de coördinaten, die daarvoor staan, aannemen. Dat is bijvoorbeeld noodzakelijk, als u het totale grafiekvenster met REC 0,0,319,199,1 wilt omkaderen. Op een ding moet u echter letten als u het effect niet met opzet wilt oproepen: geeft u geen hoogte of breedte van een rechthoek aan, die buiten het grafiekvenster zou liggen, zoals REC 100,150,50,80,1. Zoals u ziet wanneer u dit voorbeeld uitprobeert, wordt de rechthoek aan de ene (rechter-)kant correct slechts tot de rand van het beeldscherm getrokken. Aan de andere (linker-)kant echter wordt de totale hoogte getekend, die dan natuurlijk boven het oorsprongpunt van de rechthoek (linker bovenhoek) uittorent. Iets dergelijks geldt dan eveneens voor een te groot gekozen breedte. De volgende voorbeelden geven u enkele suggesties voor het gebruik van REC.

Voorbeelden: U heeft natuurlijk de mogelijkheid een kader met een bredere dikte te tekenen dan de constante dikte van 1 punt. De volgende routine maakt u deze variatie van de REC-instructie aanschouwelijk. De gewenste dikte wordt van te voren (hier in regel 170) -naast de andere noodzakelijke parameters- in het geheugen D1 opgeslagen. Het daarvoor vereiste algoritme (berekeningsvoorschrift), dat als onderprogramma in elk van uw programma's ingebouwd kan worden, zou er dan als volgt uit zien (vanaf regel 280):

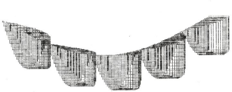
```
100 REM ****************************
110 REM **                        **
120 REM **    VARIABELE RAAMDIKTE **
130 REM **                        **
140 REM ****************************
150 REM
160 HIRES 1,0
170 DI = 9 : X = 100 : Y = 50 : B = 90 : H = 80 : REM VOORBEELD
    PARAMETERINVOER
180 GOSUB 280 : REM OPROEP SUBROUTINE
190 WAIT 198,255
200 END
210 REM
220 REM  **************************
230 REM  **                      **
240 REM  **   START SUBROUTINE   **
250 REM  **                      **
260 REM  **************************
270 REM
280 FOR Z=0 TO DI-1
290 REC X+Z, Y+Z, B - 2*Z, 80 - 2*Z,1
300 NEXT Z
310 RETURN : REM EINDE SUBROUTINE
```

U ziet, het resultaat mag er wezen. Als altijd moet u de afzonderlijke parameters veranderen (bijvoorbeeld DI).

In aansluiting hierop enige effecten, die u kunt bereiken met de rechthoekinstructie:

```
100 REM ********************
110 REM **                **
120 REM **  REC-VOORBEELD **
130 REM **                **
140 REM ********************
150 REM
160 REM    ****************
170 REM    **            **
180 REM    **  FIGUUR 1  **
190 REM    **            **
200 REM    ****************
210 REM
220 HIRES 3,4
230 FOR X=1 TO 99 STEP 2
240 REC X,X,X,X,1
250 NEXT X
260 PAUSE 10
270 HIRES 3,4
280 REM
290 REM    ****************
300 REM    **            **
310 REM    **  FIGUUR 2  **
320 REM    **            **
330 REM    ****************
340 REM
350 FOR X=1 TO 290 STEP 3
360 A = X
370 B = 60 * SIN(X/90+3.14) + 100
380 C = ABS(B/4)+10
390 D = C
400 REC A,B,C,D,1
410 NEXT X
420 PAUSE 10
430 HIRES 3,4
440 REM
450 REM    ****************
460 REM    **            **
470 REM    **  FIGUUR 3  **
480 REM    **            **
490 REM    ****************
500 REM
510 FOR X=1 TO 290 STEP 2
520 A = X
530 B = 60 * SIN(X/90) + 50
540 C = ABS(30 * SIN(X/20))
550 D = C*2
560 REC A,B,C,D,1
570 NEXT X
580 WAIT 198,255 '
```

Laat u niet teveel door de gecompliceerde berekeningen afschrikken. U hoeft het niet allemaal te begrijpen.



12.3.5 BLOCK

Format	BLOCK x1,y1,x2,y2,tm
Parameter	x1: x-coördinaat linker bovenhoek van het veld
	MC: 0-159 / HGR: 0-319
	y1: y-coördinaat linker bovenhoek van het veld
	MC: 0-199 / HGR: 0-199
	x2: x-coördinaat van de rechter benedenhoek van het veld
	MC: 0-159 / HGR: 0-319

	y2: y-coördinaat van de rechter benedenhoek van het veld
	MC: 0-199 / HGR: 0-199
	tm: tekenmodus
Voorbeeld	BLOCK 100,150,160,200,1
Functie	Tekenen van een gevulde rechthoek (veld)

Neem nu eens aan, dat u een statistiekprogramma wilt maken en daarbij een paar balkdiagrammen wilt gebruiken. U heeft de mogelijkheid om door middel van de onder REC als eerste voorbeeld aangegeven routine zo'n balk te maken, door het daar gemaakte kader zoveel dikte te geven, dat twee kaderranden elkaar raken. Zoals u zich kunt voorstellen, wordt er zeer veel tijd verknoeid. Bovendien heeft u hiervoor in BASIC een hele subroutine nodig, wat u weer geheugenruimte kost. Kortom, we hebben een instructie nodig, die zo'n opgevulde rechthoek gemakkelijk en snel tekent. Deze eis wordt door de invoering van de BLOCK-instructie ingewilligd.

Hoe is deze BLOCK-instructie te gebruiken? Dat is niet moeilijk. U neemt de linker bovenhoek van het gewenste veld, en zet de coördinaten op de eerste twee plaatsen van de in totaal vijf parameters van deze instructie, als begrenzing naar links en naar boven (bekend van de voorafgaande REC). Nu bepaalt u de grootte en vorm van het veld en plaatst de coördinaten van de linker benedenhoek van de figuur als derde en vierde parameter. OPGEPAST! Hier is een verschil met de REC-instructie. Bij de laatste geeft u niet, zoals hier, de coördinaten van de tweede hoek, maar de breedte en de hoogte direkt aan. Zoals altijd geeft de laatste parameter de teken-modus aan.

Een bijzonderheid van de BLOCK-instructie moet hier nog worden vermeld. Als u probeert de kleur, waarmee u een blok wilt tekenen, door LOW COL te veranderen, zult u weinig succes hebben. U kunt met BLOCK enkel en alleen in die kleuren tekenen, die u in het begin met HIRES hebt gekozen of die u door het tekenen onder LOW COL veranderd hebt. Voor een tussentijdse kleurbepaling moet u toch weer naar de routine met de REC-instructie terugkeren. Desondanks is de BLOCK-instructie een aantrekkelijke methode om snel en elegant velden op uw beeldscherm te brengen.

Voorbeelden:

Eerst geven we u een suggestie om een gedefinieerde dikte op het beeldscherm te tekenen, daarna volgen enkele andere aanwijzingen.

```
100 REM **********************
110 REM **                  **
120 REM **  BLOCK-VOORBEELD **
130 REM **                  **
140 REM **********************
150 REM
160 REM      **************
170 REM      **          **
180 REM      **  VENSTER **
190 REM      **          **
200 REM      **************
210 REM
220 HIRES 7,0 : REM GRAFIEK AAN
230 REM VENSTER
240 BLOCK 110,60,220,120,1 : REM BUITENSTE VELD TEKENEN
250 BLOCK 120,70,210,110,0 : REM BINNENSTE VELD WISSEN
260 PAUSE 10
270 REM
280 REM  *******************
290 REM  **               **
300 REM  **  INVERTEREN   **
310 REM  **               **
320 REM  *******************
330 REM
340 HIRES 7,0 : REM GRAFIEK WISSEN
350 FOR X=1 TO 280 STEP 20
360 BLOCK X, 50, X*1.2, 100 * SIN(X/80) + 50, 2 : REM INVERTEREN V/D
    BLOKKEN
370 NEXT X : REM VOLGENDE BLOK
380 PAUSE 10
390 REM
400 REM  ******************
410 REM  **              **
420 REM  **  MULTICOLOR  **
430 REM  **              **
440 REM  ******************
450 REM
460 HIRES 7,0
470 COLOUR 2,0 : REM VENSTER ROOD / ACHTERGRONDKLEUR = ZWART
480 MULTI 3,4,5 : REM MULTICOLOR INSCHAKELEN
490 BLOCK  5, 20,150,190,1 : REM KLEUR 1
500 BLOCK 45, 70,150,190,2 : REM KLEUR 2
510 BLOCK 85,120,150,190,3 : REM KLEUR 3
520 BLOCK 20, 50,100,150,4 : REM INVERTEREN
530 WAIT198,255 : REM WACHTEN OP TOETSAANSLAG
```

12.3.6 CIRCLE

Format	CIRCLE mx, my, rx, ry, tm
Parameter	mx: x-coördinaat van het middelpunt van de ellips (cirkel)
	MC: 0-159 / HGR: 0-319
	my: y-coördinaat van het middelpunt van de ellips (cirkel)
	MC: 0-199 / HGR: 0-199
	rx : radius van de ellips (cirkel) in x-richting (verticaal)
	MC: 0-159 / HGR: 0-319
	ry : radius van de ellips (cirkel) in y-richting (horizontaal)
	tm : tekenmodus
Voorbeeld	CIRCLE 150,80,60,30,1
Functie	Tekenen van een ellips of een cirkel

Met de CIRCLE-instructie kunt u cirkels en ellipsen op uw scherm toveren, door aan uw computer enkele parameters te geven, die de ellipsen nader bepalen. Eerst iets over de geometrische principes die u in dit verband over cirkels en ellipsen moet kennen. Ten eerste zijn er verschillende manieren, om een ellips te definiëren. In Simon's BASIC wordt vanwege de eenvoud het middelpunt gebruikt voor de plaatsbepaling. Belangrijk voor de definitie van onze ellipsen zijn nu de twee speciale radii in verticale en horizontale richting. Denkt u zich in, dat u een verticale streep van het middelpunt naar de rand trekt, dan krijgen we de eerste (verticale) radius. Trekt u een horizontale streep van het middelpunt naar de rand, dan krijgt u de tweede (horizontale) radius. Deze beide radii zijn bij een ellips steeds verschillend. De cirkel is een speciaal geval van een ellips: de beide radii zijn gelijk. Dus geeft u met de CIRCLE-instructie allereerst de coördinaten aan van het middelpunt van de door u gewenste ellips (mx, my). Daarna bepaalt u de verticale en horizontale maat van de ellips, door de x- (=verticale) en y- (=horizontale) radius te geven (rx, ry). Als u dit gedaan hebt, voegt u nog de tekenmodus toe.

Als u een cirkel wilt tekenen, moet u zich eerst voor de geest halen, of u in de multicolourmodus of in HGR bent. In HGR hoeft u daarvoor slechts de beide radii even groot te laten worden. In MC daarentegen is immers elk punt dubbel zo breed en hoog. Hier moet u dus de x-radius half zo groot kiezen als de y-radius. Er kunnen verschillende kleine vertekeningen optreden bij verschillende monitoren. Mocht uw cirkel een keer boven het beeldscherm uitkomen, dan wordt er bij kleine overschrijdingen een streep door de als laatste in het beeld passende punt naar het daarna weer opduikende punt getrokken. Bij grote overschrijdingen onstaat er min of meer een chaos.

Voorbeelden:

Het eerste voorbeeld is om te oefenen. Dit is raadzaam op grond van de vele parameters, die u in uw instructie moet aangeven. Typ dit voorbeeld eens compleet met REM-regels. Zo kunt u gemakkelijk de afzonderlijke functies van de parameters overzien. Verander dan de eerste vier parameters na de instructie in regel 230 en bekijkt u dan het resultaat.

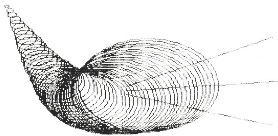
```
100 REM ************************
110 REM **                    **
120 REM **  CIRCLE-VOORBEELD 1 **
130 REM **                    **
140 REM ************************
150 REM
160 HIRES 7,8 : REM GRAFIEK AAN
170 REM VOORBEELD
```

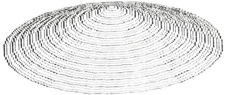
```
180 REM VERANDERT U DE PARAMETERS EENS!
190 REM
200 REM     MX! MY! RX! RY!TM
210 REM    ----+---+---+---+---
220 REM        !   !   !   !
230 CIRCLE 160,100, 70, 90, 1
240 REM    ----+---+---+---+---
250 WAIT 198,255 : REM WACHTEN OP TOETSAANSLAG
```

```
100 REM ************************
110 REM **                    **
120 REM ** CIRCLE-VOORBEELD-2 **
130 REM **                    **
140 REM ************************
150 REM
160 REM     ***************
170 REM     **           **
180 REM     **  TRECHTER  **
190 REM     **           **
200 REM     ***************
210 REM
220 HIRES 3,2 : REM GRAFIEK AAN
230 COLOUR 2,2 : REM VENSTER = ROOD
240 FOR X=1 TO 90 STEP 2
250 CIRCLE X+10,X+10,X,X,1
260 NEXT X
270 PAUSE 20
280 REM
290 REM     ***************
300 REM     **           **
310 REM     ** ALPENHOORN **
320 REM     **           **
330 REM     ***************
340 REM
350 HIRES 3,2
360 FOR X=1 TO 110 STEP 2
370 CIRCLE 1.3*X + 10, 50 * SIN(X/30 -.2) + X + 10, X/2 + X/8, X/2,1
380 NEXT X
390 LINE 130,105,270, 50,1
400 LINE 130,110,280,100,1
410 LINE 130,115,270,150,1
420 PAUSE 20
430 REM
440 REM     **************
450 REM     **          **
460 REM     **   SCHELP  **
470 REM     **          **
480 REM     **************
```

```
490 REM
500 HIRES 3,2
510 COLOUR 12,12 : REM ACHTERGROND + VENSTERKLEUR GRIJS 2
520 MULTI 6,7,7 : REM MULTICOLOR AAN
530 FOR X=1 TO 78 STEP 2
540 T = T+1 : REM KLEURREGISTER WISSELEN
550 LOW COL F,F+1,F+2 : REM VERANDERING VAN KLEUR
560 CIRCLE 80, 0.92*X+10, X, X, T
570 IF T=3 THEN T = 0 : F = F+3 : IF F=16 THEN F = 0 : REM KLEUR
    VERHOGEN
580 NEXT X
590 PAUSE 5
600 REM *******
610 REM FLITSEN
620 REM *******
630 FOR X=0 TO 15
640 COLOUR X,X : REM ACHTERGROND + VENSTERKLEUR WISSELEN
650 PAUSE 4
660 NEXT X
670 COLOUR 9,9
680 WAIT 198,255 : REM WACHTEN OP TOETSAANSLAG
```

Stoort u zich er maar niet aan, als de uitvoering van deze voorbeelden wat tijd in beslag neemt. De in Simon's BASIC ingebouwde CIRCLE-instructie is niet zo snel als in speciale grafiekprogramma's. Desondanks zult u juist van deze instructie veel plezier hebben.





12.3.7 ARC

Format	ARC mx, my, sw, ew, a, rx, ry, tm
Parameter	mx: x-coördinaat van het middelpunt van de ellips, waarvan de boog getekend moet worden
	MC: 0-159 / HGR: 0-319
	my: y-coördinaat van het middelpunt van de ellips, waarvan de boog getekend moet worden
	MC: 0-199 / HGR: 0-199
	sh: starthoek van de boog (0-360 graden)
	eh: eindhoek van de boog (0-360 graden)
	a : afstand van de hoek tussen twee te berekenen boogpunten (0-360 graden)
	rx: radius van de ellips (cirkel) in x-richting (verticaal), waarvan de boog getekend moet worden
	MC: 0-159 / HGR: 0-319
	ry: radius van de ellips (cirkel) in y-richting (horizontaal), waarvan de boog getekend moet worden

MC: 0-199 / HGR: 0-199

tm: tekenmodus

Voorbeeld ARC 160,100,20,120,20,50,40,1

Functie Tekenen van de boog van een ellips of een cirkel

Naast CIRCLE zijn er nog twee andere instructies, die zich met verschillende aspecten van de ellipsen bezighouden. De eerste van deze twee tekent slechts gedeelten van een ellips, de ellipsenbogen. We kenden al de begrippen half- of kwartcirkel; hier zijn er bijvoorbeeld halve, kwart en achtste ellipsen. Tegelijk heeft deze instructie nog een andere bijzonderheid, waarop we nog terug zullen komen. Wat is er nodig om een boog te definiëren? Eerst moet natuurlijk bekend zijn, van welke ellips (cirkel) we een boog willen tekenen. Daarmee staan we voor de taak in ieder geval eerst alle parameters aan te geven, die deze ellips (waarvan we immers alleen een gedeelte tekenen) duidelijk bepalen. De volgorde van deze parameters wordt in de bovenstaande formatbeschrijving uiteengezet. Daarnaast echter moeten we nu ook nog het bereik van de boog aangeven. Zoals u weet, kan men elke cirkel (en natuurlijk ook elke ellips) in 360 delen of graden onderverdelen. In Simon's BASIC ligt hierbij het nulpunt (dus 0 graden) op de bovenste punt van onze ellips. Van daaruit worden volgens de wijzers van de klok 360 graden uitgezet, zodat we uiteindelijk weer bij het vertrekpunt uitkomen. Als we de boog willen bepalen, geven we het aantal graden aan, waarbij de boog moet beginnen om zijn ligging te definiëren (sh) en daarna geven we het aantal graden op, waarbij de boog moet eindigen (eh). Een halfellips met het startpunt op de bovenste punt van de figuur kunnen we dus met de parameters sh=0 en eh=180 tekenen. Een kompasroos met het noorden naar boven verduidelijkt de indeling van de graden:

```
            0

     315         45

270                   90

     225        135

           180
```

Daarmee zouden we eigenlijk alles aangegeven hebben wat er nodig is voor het tekenen van een ellipsboog. Simon's BASIC gaat hier echter verder. Het is immers logisch, dat voor het tekenen van een ellips een heleboel punten berekend moeten worden. U kunt snelheid winnen

door steeds enige punten over te slaan. Normaliter gaat Simon's BASIC zo te werk, dat het achter elkaar de noodzakelijke punten in een bepaalde staplengte berekent en deze dan met behulp van een streep met elkaar verbindt, zodat er een samenhangend geheel uit onstaat. Deze staplengte is bij CIRCLE zo te kiezen, dat er een aanvaardbaar compromis wordt gevonden. Hoe verder de berekende punten uit elkaar liggen, des te hoekiger wordt de ellips en hoe meer punten er berekend worden, des te langzamer wordt er getekend. Er wordt dus in werkelijkheid een veelhoek getekend, die op ons als een ellips overkomt. Tegelijkertijd echter krijgt u daardoor de mogelijkheid een willekeurige gelijkzijdige veelhoek of gedeelten daarvan te tekenen. Daarbij geeft de daarvoor verantwoordelijke parameter a (voor afstand) steeds de afstand tussen twee punten in graden (overeenkomend met de uiteenzettingen van boven) aan. Een normale, met CIRCLE getekende ellips wordt daarbij met a=12 getekend. Dat wil zeggen: als het eerste punt bij 0 graden ligt, wordt de volgende voor 12, daarna voor 24 graden enzovoort berekend. De twee volgende instructies komen wat betreft hun resultaat aldus overeen:

```
ARC     160,100,0,360,12,50,70,1
CIRCLE  160,100,          50,70,1
```

U heeft de mogelijkheid om verschillende veelhoeken te tekenen. De volgende tabel kan u daarbij tot steun zijn:

a	figuur
120	3-hoek
90	4-hoek
72	5-hoek
60	6-hoek
51	7-hoek
45	8-hoek
40	9-hoek
36	10-hoek

Uit deze tabel kunt u dus de waarden voor a halen, die voor de afzonderlijk aangegeven figuren uit te kiezen zijn. Als u een volledige veelhoek wilt tekenen, moet u gelijktijdig sh=0 en eh=360 plaatsen. Maar dat moet u allemaal maar uitproberen in het oefenvoorbeeld.

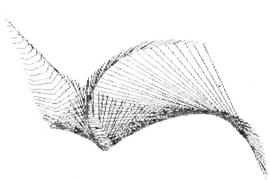
Voorbeelden:

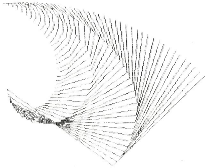
Het volgende programma is weer een oefenvoorbeeld, waarop u de uitwerking van de afzonderlijke parameters kunt toetsen.

```
100 REM **********************
110 REM **                  **
120 REM **  ARC-VOORBEELD 1 **
130 REM **                  **
140 REM **********************
150 REM
160 HIRES 7,8 : REM GRAFIEK AAN
170 REM VOORBEELD
180 REM VERANDERT U DE PARAMETERS EENS!
190 REM
200 REM  MX! MY! SH! EH!  A! RX! RY!TM
210 REM ---+---+---+---+---+---+---+--
220 REM    !   !   !   !   !   !   !
230 ARC 160,100,  0,360, 45, 70, 90, 1
240 REM ---+---+---+---+---+---+---+--
250 WAIT 198,255 : REM WACHTEN OP TOETSAANSLAG
```

```
100 REM **********************
110 REM **                  **
120 REM **  ARC-VOORBEELD 2 **
130 REM **                  **
140 REM **********************
150 REM
160 REM     ***************
170 REM     **           **
180 REM     **  FIGUUR 1 **
190 REM     **           **
200 REM     ***************
210 REM
220 HIRES 3,9 : REM GRAFIEK AAN
230 COLOUR 9,9 : REM VENSTER = BRUIN
240 FOR X=1 TO 90 STEP 2
250 ARC X+10, X+10, X, 2*X + 90, X, X, X, 1
260 NEXT X
270 PAUSE 20
280 REM
290 REM     ***************
300 REM     **           **
310 REM     **  FIGUUR 2 **
320 REM     **           **
330 REM     ***************
340 REM
350 HIRES 3,9
360 FOR X=1 TO 160 STEP 2
370 ARC 1.3*X + 10, 50 * SIN(X/30 -.2) + X+10, X, 190+X, 30+X, X/2
    + X/8, X/2, 1
```

```
380 NEXT X
390 PAUSE 20
400 REM
410 REM     ****************
420 REM     **            **
430 REM     **  FIGUUR 3  **
440 REM     **            **
450 REM     ****************
460 REM
470 HIRES 3,9
480 COLOUR 12,12 : REM ACHTERGROND- + VENSTERKLEUR GRIJS 2
490 MULTI 6,7,7 : REM MULTICOLOR AAN
500 FOR X=1 TO 78 STEP 2
510 T = T+1 : REM KLEURREGISTER WISSELEN
520 LOW COL F,F+1,F+2 : REM VERANDERING VAN KLEUR
530 ARC 80, 0.92*X+10, 90+X, 270+X, 60, X, X, T
540 IF T=3 THEN T = 0 : F = F+3 : IF F=16 THEN F = 0 : REM KLEUR
    VERHOGEN
550 NEXT X
560 PAUSE 5
570 REM *******
580 REM FLITSEN
590 REM *******
600 FOR X=0 TO 15
610 COLOUR X,X : REM ACHTERGROND- + VENSTERKLEUR VERANDEREN
620 PAUSE 4
630 NEXT X
640 COLOUR 9,9
650 WAIT 198,255 : REM WACHTEN OP TOETSAANSLAG
```





12.3.8 ANGL

Format — ANGL mx, my, w, rx, ry, tm

Parameter — mx: x-coördinaat van het middelpunt van de ellips, waarvan de radius getekend moet worden

MC: 0-159 / HGR: 0-319

my: y-coördinaat van het middelpunt van de ellips, waarvan de radius getekend moet worden

MC: 0-199 / HGR: 0-199

w : positie van de hoek van de radius 0-360 graden

rx: radius van de ellips (cirkel) in x-richting (verticaal), waarvan de radius getekend moet worden

MC: 0-159 / HGR: 0-319

ry: radius van de ellips (cirkel) in y-richting (horizontaal), waarvan de radius getekend moet worden

MC: 0-199 / HGR: 0-199

tm: tekenmodus

Voorbeeld — ANGL 160, 100,20,60,70,1

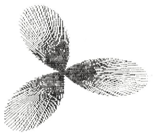
Functie		Tekenen van de radius van een ellips of een cirkel in een gedefinieerde positie van de hoek

Nu naar de beloofde andere instructie, die zich met de variaties omtrent het thema ellips bezighoudt. Wederom moet u op z'n minst de beide voorafgaande instructies gelezen hebben, om hiermee te kunnen werken. De instructie ANGL maakt het u mogelijk een willekeurige radius op een even willekeurige plaats in een (denkbeeldige) ellips te tekenen. Dat wil zeggen: niet die ellips of gedeelten daarvan worden getekend, maar alleen de radius op de aangegeven plaats. Zoals ook bij ARC moet hier eerst even de ellips, waarop de instructie betrekking heeft, gedefinieerd worden. Dat gebeurt natuurlijk weer op de oude, bekende manier met opgave van het middelpunt en de radius (mx,my....,rx,ry). Maar dat is het oude verhaaltje voor u, nietwaar? Belangrijk is nu de opgave, op welke plaats in de ellips deze radius getekend moet worden. Hier komt weer de van de ARC-instructie bekende definitie van pas. U geeft aan de computer in uw instructie dus het aantal graden dat immers overeenkomt met een plaats op de rand van de ellips. En dat is alles. Zoals geen andere instructie is deze voor het produceren van mooie effecten nuttig. U kunt er bijvoorbeeld de erbij horende ellips of een overeenkomstige veelhoek omheen tekenen.

Voorbeelden:

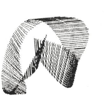
U heeft hier de mogelijkheid met behulp van het volgende oefenvoorbeeld alle parameters op hun effecten te toetsen.

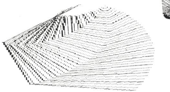
```
~~PRESS PLAY ON TAPE~~ ?

100 REM **********************
110 REM **                  **
120 REM **  ANGL-VOORBEELD 1 **
130 REM **                  **
140 REM **********************
150 REM
160 HIRES 7,8 : REM GRAFIEK AAN
170 REM VOORBEELD
180 REM VERANDERT U DE PARAMETERS EENS!
190 REM
200 REM   MX! MY!  H! RX! RY!TM
210 REM ----+---+---+---+---+---
220 REM     !   !   !   !   !
230 ANGL 160,100, 45, 70, 90, 1
240 REM ----+---+---+---+---+---
250 WAIT 198,255 : REM WACHTEN OP TOETSAANSLAG
```

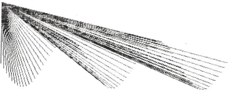


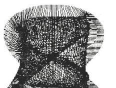
```
100 REM ***********************
110 REM **                   **
120 REM **  ANGL-VOORBEELD 2 **
130 REM **                   **
140 REM ***********************
150 REM
160 REM
170 REM      ****************
180 REM      **            **
190 REM      **  FIGUUR 1  **
200 REM      **            **
210 REM      ****************
220 REM
230 HIRES 7,8 : REM GRAFIEK AAN
240 FOR X=1 TO 140 STEP 2
250 ANGL X,X,X,X,X,1
260 NEXT X
270 REM
280 REM      ****************
290 REM      **            **
300 REM      **  KLAVERBLAD **
310 REM      **            **
320 REM      ****************
330 REM
340 HIRES 7,8 : REM GRAFIEK WISSEN
350 FOR X=1 TO 360 STEP 2
360 A = 40 * SIN(X/19 -.5) + 40
370 ANGL 160,100,X,A,A,1
380 NEXT X
390 PAUSE 20
400 REM
410 REM      ****************
420 REM      **            **
```

```
430 REM      **  FIGUUR 3  **
440 REM      **            **
450 REM      ****************
460 REM
470 HIRES 7,8 : REM GRAFIEK WISSEN
480 FOR X=1 TO 400 STEP 2
490 MX = 50 * SIN(X/60) + 100
500 MY = 60 * SIN(X/40) +  70
510 H  = X/2
520 RX = 20 * SIN(X/20) +  30
530 RY = 30 * SIN(X/30) +  30
540 ANGL MX,MY,H,RX,RY,1
550 NEXT X
560 PAUSE 20
570 REM
580 REM     *****************
590 REM     **             **
600 REM     **  MARGRIETJE **
610 REM     **             **
620 REM     *****************
630 REM
640 HIRES 7,8 : REM GRAFIEK WISSEN
650 FOR Y=0 TO 31
660 T = ABS(T-1)
670 FOR X=1 TO 20 STEP 1
680 ANGL 160, 100, X+Y*11, X*3, X*3, T
690 NEXT X
700 NEXT Y
710 WAIT 198,255
```









12.3.9 PAINT

Format	PAINT x, y, tm
Parameter	x : x-coördinaat van het testpunt
	MC: 0-159 / HGR: 0-319
	y : y-coördinaat van het testpunt MC: 0-199 / HGR: 0-199
	tm : tekenmodus
Voorbeeld	PAINT 100,120,1
Functie	Opvullen van een willekeurig omsloten vlak

De laatste in deze paragraaf te behandelen grafiekinstructie is de PAINT-instructie. De ontwerper van Simon's BASIC heeft daarmee iets fraais uitgevonden. Stel, u wilt niet alleen met de inmiddels bekende CIRCLE-instructie een eenvoudige cirkel of ellips tekenen, maar deze

ook opvullen. Of u hebt een mooi omkaderd figuur in gedachten en wilt daarmee hetzelfde doen. In dat geval gaat u als volgt te werk: u tekent eerst de omkadering van de figuur op het beeldscherm. Daarbij let u erop, dat er geen opening in deze rand voorkomt. Nu bepaalt u een enkel punt, dat binnen deze figuur ligt en geeft daarvan de coördinaten op de bekende manier aan de PAINT-instructie. De rest gaat vanzelf. PAINT vindt de omkadering en vult de figuur net punten geheel op. Het omgekeerde geldt bij het uitwissen: dan herkent PAINT alle uitgewiste punten als kader en wist eveneens alle punten die daar binnen liggen. Natuurlijk kunt u ook een punt buiten de figuur aangeven. In dat geval wordt het totale beeldscherm behalve alle afgesloten figuren opgevuld (gewist). Alleen het inverteren (tm=2) levert om onbekende redenen vreemde effecten op, die op zichzelf wel de moeite van het bekijken waard zijn. Het kan wel eens voorkomen, dat bepaalde gedeelten van figuren, vooral in spitse hoeken, niet opgevuld worden. Dat ligt echter niet aan de PAINT-instructie, maar aan de manier waarop lijnen en figuren op het beeldscherm gebracht worden.

Anders dan in de BLOCK-instructie is bij PAINT de LOW COL-instructie volledig werkzaam. U kunt dus - als u dat wilt - ter vervanging van BLOCK een rechthoek (met REC getekend) met PAINT opvullen en u heeft zo de mogelijkheid de kleur te veranderen (zie ook REL en BLOCK). Ter demonstratie weer enkele voorbeelden:

```
100 REM **********************
110 REM **                  **
120 REM **  PAINT-VOORBEELD **
130 REM **                  **
140 REM **********************
150 REM
160 REM
170 REM        ************
180 REM        **        **
190 REM        **  OVAAL **
200 REM        **        **
210 REM        ************
220 REM
230 HIRES 8,9 : REM GRAFIEK AAN
240 CIRCLE 160,100,50,80,1 : REM ELLIPS  ALS BEGRENZING
250 PAUSE 1
260 PAINT 160,100,1 : REM OPVULLEN
270 PAUSE 5
280 REC 10,10,60,70,1 : REM RECHTHOEK ALS BEGRENZING
290 PAUSE 1
300 PAINT 11,11,1
310 PAUSE 15
320 REM
```

```
330 REM     ****************
340 REM     **            **
350 REM     **  FIGUUR 2  **
360 REM     **            **
370 REM     ****************
380 REM
390 HIRES 8,9 : REM GRAFIEK WISSEN
400 REM
410 REM ***************
420 REM VELD BEGRENZEN:
430 REM ***************
440 REM
450 LINE 100,100, 50, 70,1
460 LINE  50, 70,150,120,1
470 LINE 150,120,120, 20,1
480 LINE 120, 20,300,199,1
490 LINE 300,199,160,100,1
500 LINE 160,100,160,170,1
510 LINE 160,170, 60,130,1
520 LINE  60,130,100,100,1
530 PAUSE 1
540 PAINT 101,101,1 : REM VELD INKLEUREN
550 PAUSE 20
560 PAINT 101,101,0 : REM VELD WISSEN
570 WAIT 198,255 : REM WACHTEN OP TOETSAANSLAG
```

```
100 REM **********************
110 REM **                  **
120 REM **  PAINT-VOORBEELD **
130 REM **                  **
140 REM **********************
150 REM
160 HIRES 3,2 : REM GRAFIEK AAN
170 CIRCLE 160,100,30,40,1 : REM BINNEN
180 CIRCLE 160,100,50,60,1 : REM MIDDEN
190 CIRCLE 160,100,70,80,1 : REM BUITEN
200 PAINT 160+29,100+39,1 : REM MIDDELSTE STREPEN INKLEUREN
210 PAINT 0,0,1 : REM BUITENSTE VELD INKLEUREN
220 PAUSE  15
230 REM
240 REM *************
250 REM **         **
260 REM **  BLOEM  **
270 REM **         **
280 REM *************
290 REM
300 HIRES 3,2
310 CIRCLE 160,100,10,10,1
```

```
320 CIRCLE 160,100,30,40,1
330 CIRCLE 160,100,50,20,1
340 PAINT     160,    100,1
350 PAINT       160,100+38,1
360 PAINT       160,100-38,1
370 PAINT 160+48,     100,1
380 PAINT 160-48,     100,1
390 WAIT 198,255
```

12.3.10 Testopdrachten

1. Welke syntax is juist?

   A) PLOT 1,0,0 :
      tekent een punt bij 0,0
   B) TEST (100,100) :
      toetst of er een punt bij het coördinaat 100,100 is geplaatst.
   C) IF TEST (100,100) = 0 THEN 90 :
      Als er geen punt bij 100,100 is geplaatst, wordt er naar regel 90 gesprongen
   D) REC 10,20,100,150 :
      tekent een rechthoek met de twee hoekpunten 10,20 en 100,150
   E) BLOCK 10,20,100,150 :
      tekent een veld met de twee hoekpunten 10,20 en 100,150
   F) ARC 160,100,0,360,12,50,70,1 :
      tekent een volledige ellips

2. U wilt de met CIRCLE 160,100,40,30,1 getekende ellips opvullen. Welke instructie gebruikt u?

   A) BLOCK 160,100,40,30,1
   B) PAINT 160,100,1
   C) PAINT 0,0,1
   D) BLOCK 160,100,1

3. U wilt een kader met een dikte van 10 punten tekenen. Welke

combinatie van instructies gebruikt u?

A) FOR X=0 TO 9:REC 100+X,100+X,50-2*X,60-2*X,1:NEXTX
B) BLOCK 100,100,150,160,1 : BLOCK 110,110,140,160,0
C) REC 100,100,50,60,1 : REC 110,110,30,40,1 : PAINT 105,105,1
D) REC 100,100,50,60,1

12.4 Zelf figuren tekenen

12.4.1 DRAW

Format	DRAW str, x, y, tm
Parameter	str : stringvariabele of "reeks tekens" bestaande uit de figuurdefinitie
	x : x-coördinaat van het beginpunt van de DRAW-figuur (HGR: 0-319 / MC: 0-159)
	y : y-coördinaat van het beginpunt van de DRAW-figuur (HGR: 0-199 / MC: 0-199)
	tm : tekst modus
Voorbeeld	DRAW "5678",160,100,1 of DRAW A$,160,100,1
Functie	Tekenen van een figuur

Tot nu toe hebben we ons alleen beziggehouden met figuren die door een vaste programmering of door een bepaalde berekeningswijze (algoritme) al bepaald waren. We konden deze figuren slechts vergroten of vertekenen (bijvoorbeeld met CIRCLE). De grondvorm van deze figuren was al bepaald. Nu zou u zich daar niets van kunnen aantrekken en door middel van de PLOT- en LINE-instructies uit reeds gedefinieerde punten een eigen, complexe figuur kunnen maken. Deze werkwijze kost natuurlijk veel tijd en geheugenplaatsen. Als u dan ook nog een willekeurige verschuiving van uw figuur of zelfs een rotatie wilt uitvoeren, wordt er teveel gevraagd van BASIC. Dit doel wordt door de Simon's Basic-instructie DRAW volledig op een eenvoudige en vooral zeer snelle manier bereikt. Met DRAW is het mogelijk om zelf een figuur te maken. Deze figuur kan verder willekeurig op het grafiekbeeldscherm gepositioneerd en zelfs gedraaid en vergroot worden. De te maken figuur wordt door de inhoud van de met str aangemerkte parameter vastgelegd. U maakt op deze manier een string in de vorm van een stringvariabele of als "reeks tekens" achter de DRAW-instructie. Hoe kunt u zo'n string definiëren en met welk resultaat? U geeft aan, in welke van de vier mogelijke richtingen (omhoog, omlaag, naar rechts, naar links) de grafiekcursor moet bewegen met een bepaalde lengte-eenheid, die u met ROT (volgende paragraaf) kunt bepalen. Tegelijk geeft u aan of de cursor een lijn

moet tekenen of zich moet beperken tot een beweging van de ene positie naar de andere. Het gaat hier dus om 2x4 functies die aan de computer door de volgende cijfers worden meegedeeld:

```
Cijfer.....Bewegingsrichting.....Tekenen?
 0...........rechts...............nee
 1...........omhoog...............nee
 2...........omlaag...............nee
 3...........links................nee
 4...........omlaag...............nee
 5...........rechts...............ja
 6...........omhoog...............ja
 7...........omlaag...............ja
 8...........links................ja
 9........s.t.o.p.p.e.n..............
```

```
                Dus:

                1/6

            3/8     0/5

                2/7
                (4)
```

Het cijfer 9 heeft een ontsnappingsfunctie en onderbreekt de tekenacties, ook wanneer er nog andere cijfers zouden volgen. Het hoeft echter niet aan het einde van de definitie achter elke string te staan. De figuur van het cijfer wordt dus door een achter elkaar plaatsen van deze richtingaanduidingen samengesteld. De lengte wordt alleen door de maximale stringlengte (256 tekens) begrensd. Omdat u echter slechts maximaal 80 tekens op een beeldschermregel (BASIC-regel) krijgt, moet u langere definities door het volgende trucje in meerdere regels verdelen:

```
10 A$ = "12345678012345678012345678 01
234567801234567801234567801234567780"
20 A$ = A$ + "12345241367844343278 9"
```

De variabele A$ kan nu als definitiestring in de DRAW-instructie worden gebruikt.

Na str zijn er nog drie andere parameters aan te geven, om een volledige syntax te krijgen. De laatste (tm) hoeven we hier niet verder te behandelen, omdat u die al kent uit paragraaf 12.2. De parameters x en y geven u de mogelijkheid om de positie op het beeldscherm te bepalen, waarop de tekencursor begint. Vanuit dit punt begint de computer met de verwerking van de in str gegeven aanwijzingen. Een mooi gebruik van de DRAW-instructie is de aanvulling van de sprites met deze soort figuren (shapes), die door snel uitwissen en opnieuw plaatsen van een figuur over het beeldscherm

bewogen kunnen worden. Deze techniek moet u beslist leren beheersen. Het is de moeite waard.

Voorbeelden:

```
100 REM ***********************
110 REM **                   **
120 REM **  DRAW-VOORBEELD 1 **
130 REM **                   **
140 REM ***********************
150 REM
160 HIRES 6,7 : REM GRAFIEK AAN
170 ROT 0,40 : REM GEEN ROTATIE / GROOTTE 40
180 DRAW "65789",160,100,1 : REM VIERKANT TEKENEN
190 REM BOVEN  + PLOT / RECHTS + PLOT
200 REM ONDER  + PLOT / LINKS  + PLOT
210 REM EINDE
220 WAIT 198,255 : REM WACHTEN OP TOETSAANSLAG
```

Dit voorbeeldprogramma moet als experimenteel of werkprogramma dienen. U kunt hierin veranderingen aanbrengen. De regels 190-200 beschrijven de functie van de afzonderlijke cijfers in de definitiestring. Daarbij zou u de 9 weg kunnen laten.

```
100 REM      ***********************
110 REM      **                   **
120 REM      **  DRAW-VOORBEELD 2 **
130 REM      **                   **
140 REM      ***********************
150 REM
160 REM
170 REM ********************************
180 REM **                           **
190 REM **    LETTERS DEFINIEREN     **
200 REM **                           **
210 REM ********************************
220 REM
230 DATA "6666665557577778788889" : REM D
240 DATA "66666655557778888000007779" : REM A
250 DATA "000666666888000005559" : REM T
260 DATA "66666655557778888000007779" : REM A
270 DATA "" : REM SPACE
280 DATA "666666555577788880000777788889" : REM B
290 DATA "5555533333666555553333666555559" : REM E
300 DATA "5555333366666655559" : REM C
310 DATA "6665757573331115656563337779" : REM K
```

```
320 DATA "5555533333666555553333666555559" : REM E
330 DATA "6665757573331116665557778889" : REM R
340 REM
350 REM       ***********************
360 REM       **                   **
370 REM       **  TEKEN GEDEELTE   **
380 REM       **                   **
390 REM       ***********************
400 REM
410 HIRES 2,3
420 ROT 0,4 : REM GEEN ROTATIE/ GROOTTE 4
430 FOR X=1 TO 11
440 READ A$ : REM DATA INLEZEN
450 DRAW A$, X*30 - 28, 50 + 10*X, 1 : REM LETTERS TEKENEN
460 NEXT X
470 PAUSE 4
480 REM
490 REM ****************************
500 REM ** LETTERS   VERBREDEN **
510 REM ****************************
520 REM
530 RESTORE              : REM DATAPOINTER TERUGZETTEN
540 FOR X=1 TO 11        : REM LETTERLUS
550 READ A$              : REM DATA INLEZEN
560 FOR Z=0 TO 7 STEP 1 : REM BREEDTE-LUS
570  A$, X*30 - 28 + Z , 50 + 10*X + Z, 1 : REM LETTERS TEKENEN
580 NEXT Z
590 NEXT X
600 WAIT 198,255 : REM WACHTEN OP TOETSAANSLAG
```

Dit programma geeft een reeks letters met behulp van de DRAW-instructie. De afzonderlijke letters zijn in de regels 230-330 als stringdefinitie in DATA-regels vastgelegd. Uit deze regels kunnen de afzonderlijke elementen na elkaar door de instructie READ (regel 440 en 550) in een stringvariabele ingelezen worden. Elke DATA-regel geeft hier de definitie voor een bepaald teken. Vervolgens worden de aparte letters op het beeldscherm gebracht en in 8 stappen door ze enigszins verplaatst opnieuw te tekenen, dikker gemaakt. Een tip: verandert u eens de STEP-parameter in regel 560.

12.4.2 ROT

Format	ROT r, g
Parameter	r : rotatiehoek van de figuur om het beginpunt (0-7)
	g : vergrotingsfactor (0-255)
Voorbeeld	ROT 0,20
Functie	Roteren en vergroten van de DRAW-figuur

DRAW alleen bergt een grote hoeveelheid mogelijkheden in zich. ROT is echter de kroon op het geheel. Met ROT kunt u de figuur, die u wilt maken (shape), om het referentiepunt draaien. U legt uw figuur dus schuin of draait alles op z'n kop, net zoals het u uitkomt. Uw shapes kunnen capriolen maken. Er staan 8 vastgestelde draaihoeken tot uw beschikking, die als volgt zijn vastgelegd.

r.........	rotatiehoek
0.........	0 graden
1.........	45 graden
2.........	90 graden
3.........	135 graden
4.........	180 graden
5.........	225 graden
6.........	270 graden
7.........	315 graden

Als u dus r=0 kiest, dan wordt uw met de DRAW-instructie gedefinieerde figuur getekend zoals u deze ingevoerd hebt. Maar als r bijvoorbeeld 1 is, dan wordt de figuur 45 graden volgens de wijzers van de klok gedraaid. Als u voor r een getal groter dan 7 aangeeft, verschijnt de melding BAD MODE, zoals we die al vanuit verscheidene

andere instructies kennen. Bij het gebruik van ROT voor de rotatie is er nog wat op te merken. Een omwenteling om het referentiepunt door de instructie ROT houdt niet onvoorwaardelijk de proporties van de gewentelde figuur vast. Bij oneven draaiposities (r=1,3,5,7) verschijnt de totale figuur vergroot. Dit ligt aan het feit, dat bij de omwenteling een punt niet de baan van een cirkel aflegt, maar die van een vierkant. Dit effect wordt het beste in het volgende voorbeeld duidelijk.

```
10 HIRES 2,3 : REM GRAFIEK AAN
20 FOR R=0 TO 7
30 PAUSE 2
40 ROT R,20 : REM ROTEREN
50 DRAW "6",160,100,1 : REM STREEP TEKENEN
60 NEXT R
70 WAIT 198,255: REM WACHTEN OP TOETSAANSLAG
```

Naast r kunt u nog een tweede parameter aangeven: g. Met g bepaalt u de factor, waarmee u uw figuur wilt vergroten. Normaliter komt elke stap naar rechts, links, boven of beneden (zie DRAW) bij de definitie van de shapes overeen met een stap van een punt in de aangegeven richting. De staplengte bedraagt een punt. Deze staplengte en daarmee de grootte van de totale figuur kunt u nu met g veranderen. De parameter g kan waarden van 0 tot 255 aannemen, waarbij 0 overeenkomt met een vergroting van 256 eenheden. Gedeelten van de tekening, die boven de rand van het beeldscherm uitkomen, worden afgesneden. Uiteraard moet u hier op de veranderde verhoudingen van de groottes en op vertekeningen in de multicolourmodus letten. Nu enkele voorbeelden, die u het gebruik van de instructies DRAW en ROT duidelijk moeten maken. Daarbij laat het eerste gedeelte u de omgang met bewegende shapes zien.

Voorbeelden:

```
100 REM ******************
110 REM **              **
120 REM **  ROT-VOORBEELD **
130 REM **              **
140 REM ******************
150 REM
160 REM
170 REM   ***************
180 REM   **           **
190 REM   **    KLOK   **
200 REM   **           **
210 REM   ***************
```

```
220 REM
230 HIRES 7,6 : REM GRAFIEK AAN
240 FOR T=1 TO 20 : REM RONDE LUS
250 FOR X=0 TO  7 : REM ROTATIE LUS
260 ROT L,40 : REM LAATSTE ROTATIEGRAAD INSTELLEN
270 DRAW "6",160,100,0 : REM LAATSTE STREEP WISSEN
280 ROT X,40 : REM NIEUWE ROTATIEGRAAD
290 DRAW "6",160,100,1 : REM STREEP TEKENEN
300 L = X : REM LAATSTE ROTATIEGRAAD OPSLAAN
310 NEXT X
320 NEXT T
330 PAUSE 3
340 REM
350 REM     ***************
360 REM     **           **
370 REM     **  BEELD 2  **
380 REM     **           **
390 REM     ***************
400 REM
410 HIRES 7,6 : REM GRAFIEK WISSEN
420 FOR X=1 TO 69
430 ROT X/10, X : REM ROTATIE EN GROOTTE
440 DRAW "365577886",160,100,1
450 NEXT X
460 PAUSE 10
470 REM
480 REM     *************
490 REM     **         **
500 REM     **  STER   **
510 REM     **         **
520 REM     *************
530 REM
540 HIRES 7,6 : REM GRAFIEK WISSEN
550 FOR X=1 TO 69
560 ROT X/10, X : REM ROTATIE EN GROOTTE
570 DRAW "5885766",160,100,1
580 NEXT X
590 WAIT 198,255 : REM WACHTEN OP TOETSAANSLAG
```

In het eerste gedeelte wordt u het omgaan met bewegingen uitgelegd. Belangrijk is daarbij, dat allereerst het object op de volgende positie getekend wordt, waarna het reeds getekende oude object op de laatste positie wordt uitgewist om een zo vloeiend mogelijke beweging te simuleren. Soms moet u ook de omgekeerde weg bewandelen om, vooral bij grote figuren, hinderlijke lijnen te elimineren. Zo kunt u niet alleen laten roteren, maar eveneens in verschillende richtingen laten bewegen, doordat u steeds op nieuwe cöu(o)rdinaten tekent. Dat moge het volgende voorbeeld duidelijk maken.

```
100 REM **********************
110 REM **                  **
120 REM **  ROT-VOORBEELD 2 **
130 REM **                  **
140 REM **********************
150 REM
160 HIRES 2,3 : REM GRAFIEK AAN
170 ROT 0,20 : REM GEEN ROTATIE
180 FOR X=1 TO 319 STEP 4
190 DRAW "6578",A,100,0 : REM OP OUDE POSITIE WISSEN
200 DRAW "6578",X,100,1 : REM OP NIEUWE POSITIE TEKENEN
210 A = X : REM OUDE POSITIE OPSLAAN
220 NEXT X
230 WAIT 198,255 : REM WACHTEN OP TOETSAANSLAG
```

12.4.3 Testopdrachten

1. Welke instructies gebruikt u, als u een kwadraat wilt tekenen?

   A) REC 160,100,70,70,1
   B) DRAW "0132",160,100,1
   C) DRAW "5687",160,100.1
   D) ROT 4,5

2. U wilt uw figuur 90 graden tegen de wijzers van de klok in draaien en vijf keer vergroten. Welke instructies gebruikt u?

   A) ROT 90,5
   B) ROT 2,5
   C) ROT 6,5
   D) ROT 5,2

3. Welke cijfertoevoegingen voor de shapedefinities zijn juist?

   A) 0 : rechts zonder tekenen
   B) 8 : rechts met tekenen
   C) 4 : links zonder tekenen
   D) 6 : boven met tekenen
   E) 9 : beneden met tekenen

12.5 Tekst in grafiek

12.5.1 CHAR

Format	CHAR x, y, c, tm, g
Parameter	x : x-coördinaat van de linker bovenhoek van het teken
	HGR: 0-319 / MC: 0-159
	y : y-coördinaat van de linker bovenhoek van het teken
	HGR: 0-199 / MC: 0-199
	c : beeldschermcode van het teken (0-255)
	tm : tekenmodus
	g : verticale grootte van het teken (0-255)
Voorbeeld	CHAR 160,100,3,1,2
Functie	Plaatsen van een karakter

Uw grafische voorstellingen kunt u met deze instructie van bijschriften voorzien. Een uitgebreide toelichting of eenvoudige, korte commentaren horen er gewoon bij. Deze spannende materie zal u niet onberoerd laten. U kunt met deze instructie een afzonderlijke letter of een willekeurig grafiekteken, normaal of inverse, op het beeldscherm brengen. Dit teken definieert u met de parameter c (voor beeldschermcode). Zoals u misschien nog weet van het 9e hoofdstuk (zie FCHR of FILL), worden de afzonderlijke letters of grafiektekens van uw Commodore 64 in het beeldschermgeheugen onder een andere code dan de ASCII-code opgeslagen. U vindt een volledige beeldschermcode-tabel in uw CBM-handboek. Om de codes van de inverse tekens te krijgen, moet u steeds 128 optellen bij de code van de normale tekens. Met behulp van deze tabel geeft u de parameter c voor het door u gekozen teken. Zoals u ziet, is er geen mogelijkheid om de te gebruiken karakterset uit te kiezen. In plaats daarvan worden slechts de tekens van de hoofdletter/grafiek-modus door deze instructie op het beeldscherm gebracht. Om de tekens van

een alternatieve karakterset uit te kiezen, moet u de hierna behandelde instructie TEXT gebruiken. U kunt nu aangeven, op welke plaats van het beeldscherm u een teken wilt zetten. Dat gebeurt met behulp van de parameters x en y, die de coördinaten van de linker bovenhoek van het teken op het beeldscherm bepalen. Als u dus de linker bovenhoek van uw teken in het nulpunt wilt positioneren, moet u x=0 en y=0 opgeven. U kunt een parameter g aan het eind van de instructie opgeven, om de vergrotingsfactor van het teken vast te leggen. Anders dan met ROT, wordt hier elk teken alleen in verticale richting vergroot. De tekens worden dus alleen langer. De parameter g kan waarden tussen 1 en 255 aannemen. Slechts de waarden van 1 tot 30 zijn zinvol. Daarna worden de tekens volkomen onleesbaar.

Zoals al bij eerdere instructies, die erg veel parameters hadden, wordt u ook hier een werkvoorbeeld gegeven, dat u zo veel mogelijk in z'n geheel (ook met REM-regels) moet vertalen, om de afzonderlijke parameters zo snel en juist mogelijk te begrijpen en te onthouden.

```
100 REM **********************
110 REM **                  **
120 REM **  CHAR-VOORBEELD-1 **
130 REM **                  **
140 REM **********************
150 REM
160 HIRES 1,0 : REM GRAFIEK AAN
170 REM CHARVOORBEELD
180 REM VERANDERT U DE PARAMETERS EENS!
190 REM
200 REM     X!  Y!  C!TM! G
210 REM ----+---+---+--+---
220 REM     !   !   !  !
230 CHAR 160,100,  2, 1, 3
240 REM ----+---+---+--+---
250 WAIT 198,255 : REM WACHTEN OP TOETSAANSLAG
```

```
100 REM **********************
110 REM **                  **
120 REM **  CHAR-VOORBEELD 2 **
130 REM **                  **
140 REM **********************
150 REM
160 REM
170 REM   *********************
180 REM   **                 **
190 REM   ** TEKENS VOORRAAD **
200 REM   **                 **
210 REM   *********************
```

```
220 REM
230 HIRES 5,6
240 FOR X=0 TO 255
250 T = X/40
260 CHAR FRAC(T) * 40 * 8, X/2, X, 1, 2
270 NEXT X
280 PAUSE 20
290 REM
300 REM      **************
310 REM      **          **
320 REM      **  GROOTTE **
330 REM      **          **
340 REM      **************
350 REM
360 HIRES 5,6
370 FOR X=1 TO 26
380 CHAR X*8, X, X, 1, X
390 NEXT X
400 WAIT 198,255 : REM WACHTEN OP TOETSAANSLAG
```

Het eerste gedeelte van het tweede voorbeeld toont u de karakterset. Het tweede gedeelte illustreert het gebruik van parameter g.

12.5.2 TEXT

Format TEXT x, y, str, tm, g, a

Parameter x : x-coördinaat van de linker bovenhoek van het eerste teken van de string

HGR: 0-319 / MC: 0-159

y : y-coördinaat van de linker bovenhoek van het eerste teken van de string

HGR: 0-199 / MC: 0-199

str : stringgeheugen of "reeks van tekens" met de uit te voeren tekens

tm : tekst modus

g : verticale grootte van het teken (0-255)

a : afstand van de linker punten van twee tekens

Voorbeeld TEXT 50,100,"TEST",1,3,15

Functie Schrijven van een tekst

Wanneer u een langere tekst of kleine letters in uw grafische voorstelling wilt aanbrengen, blijkt de CHAR-instructie te omslachtig en te moeizaam te zijn. Dan komt de TEXT-instructie van pas. U geeft hierbij met x en y de coördinaten van de linker bovenhoek van de eerste letter van de tekst-string aan. Daarna geeft u de string in de vorm van een stringvariabele, of direkt als reeks karakters. U kunt uit alle tekens van de standaard karakterset kiezen. Met CTRL-A (die in de string als een inverse A verschijnt) kiest u de hoofdletter/grafiek-modus. Als u een CTRL-B invoegt, (verschijnt als inverse B) dan is de kleine/hoofdletter-modus gekozen. U kunt dus ook binnen een string tussen de afzonderlijke delen van de karakterset omschakelen zonder de reeds geplaatste tekens te veranderen. Samengevat ziet dat er zo uit:

```
10 HIRES 6,7
20 TEXT 50,100,"(<CTRL> A)TEXT-(<CTRL> B)TEXT,1,1,8
30 WAIT 198,255: REM WACHTEN OP TOETSAANSLAG
```

Met CTRL-A bedoelen we natuurlijk het tegelijk indrukken van de CTRL-toets en de A. U kunt nog een paar parameters aan de instructie toevoegen: tm voor tekenmodus (al bekend) en g (zoals bij CHAR) voor de verticale grootte. Met een andere parameter (a) bepaalt u de afstand tussen de afzonderlijke tekens van de string op het beeldscherm. Daarbij is telkens het aantal beeldschermpunten van de linkerrand van een teken tot de linkerrand van het volgende teken bedoeld. Als u dus de afstand te klein kiest (kleiner dan 7), zullen de afzonderlijke tekens elkaar overlappen, omdat elk teken een breedte van 7 punten heeft. Met behulp van deze parameters kunt u uw teksten uit elkaar trekken. TEXT geeft u veel variatiemogelijkheden.

In de gangbare versie V2 van Simon's BASIC is het helaas niet mogelijk om bij de TEXT-instructie de parameters g en a door een variabele of een expressie te bepalen, omdat er dan gedeeltes van de uit te voeren tekst verdwijnen of er andere tekens bij komen. U mag dus alleen cijfers voor deze waarden gebruiken. Verder voldoen ze aan de verwachting.

Nu een oefenprogramma:

```
100 REM ********************
110 REM **                **
120 REM ** TEXT-VOORBEELD **
130 REM **                **
140 REM ********************
150 REM
160 HIRES 1,0 : REM GRAFIEK AAN
170 REM TEXT-VOORBEELD
180 REM VERANDER DE PARAMETERS EENS !
190 REM
200 REM    X!  Y!    STR    !TM! G! A
210 REM ----+---+-----------+--+--+---
220 REM     !   !           !  !  !
230 TEXT  50!100!"DEMO-TEXT", 1, 2, 30
240 REM ----+---+-----------+--+--+---
250 WAIT 198,255:REM WACHTEN OP TOETSAANSLAG
```

```
100 REM ***********************
110 REM **                   **
120 REM **  TEXT-VOORBEELD 2 **
130 REM **                   **
140 REM ***********************
150 REM
160 HIRES 9,10 : REM GRAFIEK AAN
170 FOR X=1 TO 8
180 TEXT 100, 25*X-20, "CBM 64",1,3,20
190 NEXT X
200 PAUSE 5
210 REM
220 REM
230 REM
240 HIRES 9,10 : REM GRAFIEK WISSEN
250 REC     4, 4,311,191,1
260 LINE    0, 0,319,199,1
270 LINE  319, 0,  0,199,1
280 BLOCK  90,80,220,120,1
290 BLOCK 100,90,210,110,0
300 TEXT  103,97,"SIMON'S BASIC",1,1,8
310 WAIT 198,255 : REM WACHTEN OP TOETS AANSLAG
```

12.5.3 Testopdrachten

1. Er moet een grote normale A op het beeldscherm op de plaats met de coördinaten 160,100 in originele grootte komen. Welke instructies gebruikt u?

   A) CHAR 160,100,128+1,1,1
   B) TEXT 160,100,"A",1,1,1
   C) CHAR 160,100,1,1,1
   D) TEXT 160,100,"A",1,2,2

2. Wat gebeurt er met de te tekenen tekens, als u in CHAR of in TEXT de parameter, die we g genoemd hebben, vergroot?

   A) De tekens worden verticaal verkleind
   B) De tekens worden in x-richting vergroot
   C) De tekens worden in y-richting vergroot
   D) De tekens worden in beide richtingen vergroot
   E) De tekens worden horizontaal verkleind
   F) Geen van deze mogelijkheden

3. Hoe kiest u de parameter a bij TEXT, als u de te tekenen letters met een tussenruimte van steeds een punt op het beeldscherm wilt brengen?

   A) a=0
   B) a=1
   C) a=7
   D) a=8
   E) a=9

13 EEN KARAKTERSET MAKEN

Naast de ongewoon rijke grafiekmogelijkheden, biedt Simon's BASIC de Commodore 64 nog meer kostbaarheden. Een daarvan is de verandering van de karakterset. Ze is de basis van bijna alle spelletjes en is ook het middel (naast de sprites), dat alle spelletjes op de CBM 64 zo snel en toch rijk gevarieerd laat worden. Zonder deze mogelijkheid is een behoorlijke spelprogrammering ondenkbaar. Waar andere computers met enorme grafiekgeheugens te maken hebben, daar knipt uw Commodore 64 even met de toetsen.

Wat is nu eigenlijk een "karakterset"? Dat is, oppervlakkig bekeken, wel eenvoudig te verklaren. Onder karakterset verstaan we de totaliteit van alle grafische en lettertekens, die u door op de verschillende toetsen en toetscombinaties te drukken, op het beeldscherm kunt brengen. De vorm en het uiterlijk van deze tekens zijn de computer natuurlijk bekend, en moeten dus op de een of andere plaats en manier opgeslagen zijn. Tegelijkertijd moeten ze natuurlijk ook volgens bepaalde criteria gedefinieerd zijn, zodat uw computer weet, dat hij bijvoorbeeld een A op het beeldscherm moet brengen, als u op toets A drukt. Dit soort informatie is in alle computers opgeslagen.

Bij de CBM 64 is de karakterset zo in het geheugen geplaatst, dat hij vanuit een programma te bereiken is, de inhoud opgelezen en bijvoorbeeld ergens in een ander geheugengebied gekopieerd kan worden. Alleen dit helpt ons natuurlijk nog niet zo veel verder. Bij de meeste computers staat de karakterset in een ROM (Read Only Memory = geheugen waaruit alleen gelezen kan worden). Door de karakterset van de Commodore 64 vanuit het ROM in een deel van het RAM (waarin ook geschreven kan worden) te verplaatsen, kunnen we daarna veranderingen in de karakterset aanbrengen. U heeft daarbij de mogelijkheid om aan uw Commodore 64 te zeggen, dat hij de tekens van nu af aan bijvoorbeeld uit het geheugenbereik vanaf $2000 (= 8192) dus uit het RAM moet halen. Deze geheugenposities kunnen we zelf bepalen. Als we nu van te voren de karakterset uit het oorspronkelijke ROM in dit bereik gekopieerd hebben, zien we eerst geen verandering, omdat immers alle informatie hetzelfde gebleven is. Veranderen we nu echter gedeelten van dit geheugenbereik, dan veranderen we daarmee gelijktijdig de vorm van de tekens. Dat brengt ons op de vraag: hoe wordt de vorm van een teken opgeslagen? U weet dat uw computer in totaal 4 delen in de karakterset met elk 128 tekens heeft. Er kunnen steeds slechts 2 van die delen gelijktijdig op het tekstbeeldscherm verschijnen. De delen zijn:

zetting A/1 - normaal - hoofdletters/grafiektekens
zetting A/2 - inverse - hoofdletters/grafiektekens
zetting B/1 - normaal - hoofdletters/kleine letters
zetting B/2 - inverse - hoofdletters/kleine letters

U kunt de beide delen A en B van de karakterset door de gelijktijdige bediening van de toetsen C= en SHIFT met de hand wisselen. Vanuit het programma dienen hiervoor de ASCII-waarden 14 en 142.

UITLEG: 142= 128+14

dat wil zeggen dat u met

```
PRINT CHR$(14);
```

op zetting B en met

```
PRINT CHR$(142);
```

op zetting A kunt overschakelen
(zie hiervoor ook de instructies CSET 0 en CSET 1).

Met PRINT CHR$(8)

blokkeert u de mogelijkheid van de overschakeling met behulp van het toetsenbord, wat immers ook tijdens het proces van een programma mogelijk is en met

```
PRINT CHR$(9)
```

doet u deze blokkade weer teniet.
De overschakeling tussen zetting 1 en 2 bewerkstelligt u door het gebruik van RVS ON en RVS OFF.

In het geheugen van de karakterset moeten natuurlijk deze vier karaktersetdelen apart op een lijst gezet zijn. U heeft dus de mogelijkheid 4*128=512 tekens te veranderen. Er kunnen daarvan slechts telkens 256 verschillende tekens gelijktijdig worden gebruikt. Elk teken op het beeldscherm bestaat uit een matrix van 8*8 punten. Daarom moeten in het geheugen van de karakterset deze in totaal 64 punten gedefinieerd zijn. Dat wordt bereikt, doordat elk punt van het teken op het beeldscherm, net als in HGR, door een bit in het geheugen vertegenwoordigd wordt. Daardoor is er een bit-patroon, bestaande uit 8 bytes van elk 8 bits. We kunnen ons de opslag van een teken als volgt voorstellen:

```
Bit      7 6 5 4 3 2 1 0
Byte 0   . . . . . . . .
Byte 1   . . . . . . . .
Byte 2   . . . . . . . .
Byte 3   . . . . . . . .
Byte 4   . . . . . . . .
Byte 5   . . . . . . . .
Byte 6   . . . . . . . .
Byte 7   . . . . . . . .
```

Het geheugen van de karakterset bevat in totaal uit 512 van deze soort achter elkaar liggende definities en heeft dus een geheugenbereik van 4 Kbytes (=4096 bytes) nodig, dat standaard in het ROM van $DOOO - $DFFF ligt. Ter demonstratie wordt hier getoond, hoe een normale, grote A in het geheugen van de karakterset gedefinieerd wordt:

```
Bit      7 6 5 4 3 2 1 0
Byte 0   . . . * * . . .
Byte 1   . . * * * * . .
Byte 2   . * * . . * * .
Byte 3   . * * * * * * .
Byte 4   . * * . . * * .
Byte 5   . * * . . * * .
Byte 6   . * * . . * * .
Byte 7   . . . . . . . .
```

We krijgen dus de volgende 8 bytes:

```
Byte 0: 00011000 = $18 = 024
Byte 1: 00111100 = $3C = 060
Byte 2: 01100110 = $66 = 102
Byte 3: 01111110 = $7E = 126
Byte 4: 01100110 = $66 = 102
Byte 5: 01100110 = $66 = 102
Byte 6: 01100110 = $66 = 102
Byte 7: 00000000 = $00 = 000
```

Deze acht waarden staan nu op de plaats die voor de normale hoofdletter 'A' gereserveerd is in het geheugen van de karakterset. Voor het veranderen van een teken moet u de positie van dat teken kennen. Uitgangspunt bij alle berekeningen zijn de beeldschermcodes van de afzonderlijke tekens, van die codes dus, die voor de bepaling van een teken in het beeldschermgeheugen staan. De rest is

betrekkelijk eenvoudig. Omdat elk teken acht bytes nodig heeft, moeten we de waarde van de beeldschermcode maal acht nemen en het basisadres van het geheugen voor de tekens, dus het beginadres, waarbij het geheugen voor de tekens begint, erbij optellen. Dan krijgen we de formule:

adres = basisadres + 8 * beeldschermcode

Voor het teken 'A' in de normale ASCII-set zou dit zijn:

adres = $D000 + 8 * 1 = $D008 = 53248

Nu heeft u een beetje inzicht in het proces van het maken van een set tekens. Verdere informatie voor wat betreft de realisering in het Simon's BASIC vindt u bij de afzonderlijke instructies.

13.1 MEM

Format	MEM
Parameter	-----
Voorbeeld	MEM
Functie	Verplaatsen van de tekensset van $D000(ROM) naar $E000(RAM)

Zoals we in de inleiding van dit hoofdstuk gezien hebben, ligt de karakterset in de oorspronkelijke toestand in 4K ROM van $D000 - $DFFF. Omdat we als einddoel hebben de definitie van enkele tekens te veranderen, moeten we hem noodzakelijk naar een RAM-bereik verplaatsen. Deze functie (en nog enkele andere) wordt door de instructie MEM uitgevoerd. MEM verplaatst de ASCII-set daarbij onder het geheugenbereik van het besturingssysteem, dus van $E000 - $EFFF. Hier ligt eveneens een gedeelte van het grafiekgeheugen ($E000 - $FFFF) wat ertoe leidt, dat niet tegelijkertijd de grafiek gebruikt en de ASCII-set veranderd kan worden. Als u dat toch doet, krijgt u vreemde effecten. Ter illustratie het volgende programma.

```
10 HIRES 6,7 : REM GRAFIEK INSCHAKELEN
20 MEM : REM ASCII-SET VERPLAATSEN EN KOPIEREN
```

```
30 WAIT 198,255 : REM WACHT OP TOETSINDRUK
```

Zoals u ziet, wordt allereerst op grafiek overgeschakeld en daarna de ASCII-set naar $E000 verschoven. De totale ASCII-set verschijnt daarmee in de grafische afbeelding, omdat de beide bereiken elkaar overlappen. Omdat de opbouw van de tekens lijkt op die van de grafiek, kunt u de tekens echt herkennen. Een ander effect komt er nog bij: nadat u het programma verlaten hebt, komt u niet - zoals anders - weer in de tekstmodus terug. Tegelijkertijd verschijnt de tekst die u intikt, als een klein kleurvierkantje op het grafiekbeeldscherm, omdat het video-RAM (dus het kleurgeheugen en tegelijk beeldschermgeheugen) door de MEM-instructie zich nu op $CC00 bevindt. Omdat Simon's BASIC in de grafiekmodus aanneemt, dat het kleurgeheugen bij $C000 (49152) ligt, kunt u de kleur niet door grafiekinstructies veranderen. Pas door CSET of NRM komt u weer terug in de tekstmodus. Samenvatend kunnen we dus zeggen, dat de instructie MEM de volgende functies uitvoert:

A. verschuiving van de ASCII-set - ROM-inhoud naar $E000
B. aan de computer wordt meegedeeld, dat de ASCII-set zich nu daar bevindt
C. verschuiving van het video-RAM (beeldscherm- of tekstgeheugen) naar $CC00
D. wissen van het beeldscherm
E. verplaatsing van het SPRITE-bereik naar $C000

Daarmee kunnen nu veranderingen in de afzonderlijke regels uitgevoerd worden.

Voorbeeld:

```
100 REM ********************
110 REM **                **
120 REM **  MEM-VOORBEELD **
130 REM **                **
140 REM ********************
150 REM
160 REM
170 REM################################
180 REM********************************
190 MEM : REM KARAKTER SET IN RAM *##
200 REM********************************
210 REM################################
220 REM
230 BA = 14*4096 : REM BASISADRESSEN ($E000)
240 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
250 REM
```

```
260 REM ########################
270 REM ##                    ##
280 REM ##   ALLE TEKENS  LISTEN  ##
290 REM ##                    ##
300 REM ########################
310 REM
320 FOR Y=0 TO 24
330 FOR X=0 TO 39 STEP 4
340 FILL Y,X,4,1,C,0
350 C = C+1 : REM CODE VERHOGEN
360 NEXT X
370 NEXT Y
380 PAUSE 2
390 REM
400 REM ###################
410 REM ##  DIT VERANDEREN :  ##
420 REM###                ####
430 REM*****            ******
440 C = 0 : REM BEELDSCHERMCODE *##
450 REM*********************##
460 REM#######################
470 REM
480 REM ##################
490 REM ##              ##
500 REM ## FLITSEN VAN TEKEN ##
510 REM ##              ##
520 REM ##################
530 REM
540 FOR X=0 TO 40
550 T = ABS(T-255) : REM TUSSEN 0 EN 255 WISSELEN
560 FOR Y=0 TO 7
570 POKE BA + C*8 + Y,T : REM STOPPEN MET FLITSEN
580 NEXT Y
590 NEXT X
600 REM
610 REM ##################
620 REM ##              ##
630 REM ##  TEKEN VERANDEREN ##
640 REM ##              ##
650 REM ##################
660 REM
670 REM                  %%%%%%%%%%%%
680 POKE BA + C*8 + 0, % 01100000 :
690 POKE BA + C*8 + 1, % 01101100 :
700 POKE BA + C*8 + 2, % 01111110 :
710 POKE BA + C*8 + 3, % 00110110 :
720 POKE BA + C*8 + 4, % 00000110 :
730 POKE BA + C*8 + 5, % 00001111 :
740 POKE BA + C*8 + 6, % 00000110 :
```

```
750 POKE BA + C*8 + 7, % 00000000 :
760 REM                 %%%%%%%%%%%
770 REM
780 WAIT 198,255 : REM WACHTEN OP TOETSAANSLAG
790 PRINT CHR$(147):BEELDSCHERM WISSEN
800 LIST
```

In dit programma wordt allereerst in regel 190 de ASCII-set verschoven om die te kunnen veranderen. Daarna worden (bijna) alle tekens van de karaktersetdelen A1 en A2 opgesomd, om veranderingen van dat ogenblik te kunnen vaststellen. Het te veranderen teken zal kort oplichten en direkt daarna schrijft het programnma de nieuw ontworpen 8 bytes (regels 680-750). Hier moet u uw eigen tekens bepalen. De nieuwe tekens worden opgeslagen door middel van de bekende adresseerformule op de juiste karaktersetlocatie, die voor het betreffende teken (variabele C bevat de beeldschermcode van het teken) verantwoordelijk is. U kunt in de regels 680-750 zelf ontworpen creaties opstellen (vergeet u niet, om na elke regel RETURN in te voeren). Voor de definitie van de tekens kunt u ook in paragraaf 13.3 kijken.

13.2 DESIGN 2

Format	DESIGN 2,ta
Parameter	ta :adres van de definitie van het te veranderen teken, te berekenen volgens de formule: ta = $E000 + B * c waarbij c = beeldschermcode van het teken
Voorbeeld	DESIGN 2,$E000 + 8 * 1
Functie	Opnieuw definiëren van een teken

Zoals u bij MEM gezien hebt, kunnen we een teken opnieuw definiëren. Iets gemakkelijker kan dat echter door twee andere instructies van Simon's BASIC, die nauw samenhangen en waarvan de eerste nu verklaard wordt. DESIGN kan pas na de instructie MEM ingevoerd worden en legt vast, welk teken opnieuw gedefinieerd moet worden. Het bepaalt door zijn positie in het programma welke definitie van tekens hiervoor gebruikt moet worden. Dat gebeurt als volgt. Met ta geeft u aan voor welk teken de nieuwe creatie moet gelden. Daarbij stelt ta het adres van het tekenpatroon in het

geheugen voor, die wordt bepaald door de ons bekende formule:

adres = beginadres + 8 * beeldschermcode

Omdat door de van te voren ingevoerde MEM-instructie de karakterset naar $E000 werd verschoven, is het beginadres $E000. De beeldschermcode hangt van het teken af. Voor de letter 'A' luidt de formule dus :

ta = $E000 + 8 * 1 = $E008

In plaats van $E000 kan natuurlijk ook het daarmee overeenkomende decimale getal aangegeven worden. Om vast te leggen welk nieuw tekenpatroon, dat in zogenaamde patroonregels (zie 13.3) aangegeven is, in plaats van de letter 'A' in de toekomst op het beeldscherm moet verschijnen, moet u de DESIGN-instructie direkt voor de definitie van het teken plaatsen, anders krijgt u de foutmelding BAD CHAR FOR A MOB. De rest van de informatie krijgt u in paragraaf 13.3.

13.3 Apestaart @

Format	apestaart...tekenpatroon ....
Parameter	- tekenpatroon: zie beneden
Voorbeeld	zie beneden
Functie	Opslaan van het tekenpatroon

Nu komen we bij de belangrijkste instructie van de herdefiniëring van de tekens, de apestaart. De apestaart heeft dezelfde functie als het DATA-statement van het normale BASIC. Hij staat telkens aan het begin van een regel (hierna "patroonregel" genoemd) en is het kenmerk voor een serie data. In dit geval is dat de definitie van het teken. Deze komt op de volgende manier tot stand. Zoals u uit de inleiding van dit hoofdstuk weet, is een teken uit een 8*8 puntmatrix samengesteld. Zo ziet ook de definitie eruit. Elk van de 8 voor een teken noodzakelijke regels wordt in een afzonderlijke BASIC-regel ingevoerd. Elke regel bevat daarbij het patroon voor de 8 punten van een tekenregel in de volgende vorm (hier voor het voorbeeld van een normale hoofdletter A):

```
100 @ ...BB...
110 @ ..BBBB..
120 @ .BB..BB.
130 @ .BBBBBB.
140 @ .BB..BB.
150 @ .BB..BB.
160 @ .BB..BB.
170 @ ........
```

U plaatst voor elke in het teken geplaatste punt een B in het tekenpatroon, daarentegen een punt voor de punten, die niet geplaatst worden. Zo krijgt u een grote afbeelding van het toekomstige teken al in de BASIC-listing. U zult in hoofdstuk 14 zien dat sprites ook zo gedefinieerd worden. Direkt voor deze definitie moet nu de DESIGN-instructie staan om te bepalen, welk teken nu het nieuwe patroon moet bevatten. In regel 90 uit het voorafgaande voorbeeld had u moeten invoeren:

```
90 DESIGN 2, $EOOO + C*8
```

waarbij C de beeldschermcode is. Om de zaak compleet te maken, voegt u regel 80 toe:

```
80 MEM
```

Daarmee wordt de karakterset naar $E000 verschoven. Gebruikt u een ander patroon in de regels 100 t/m 170, dan is uw teken opnieuw gedefinieerd als u in dit voorbeeld C=1 kiest. Laat u het patroon ongewijzigd gelden voor de hoofdletter "A", dan verandert er natuurlijk niets. In plaats van de vele punten in een patroonregel kunt u ook spatietekens plaatsen, waarbij echter gegarandeerd moet zijn, dat minimaal een van de twee laatste tekens van een regel geen spatieteken is, omdat er anders BAD CHAR FOR A MOB als foutmelding opgegeven wordt. Dat gebeurt ook als er een fout teken of teveel tekens in een regel staan. Met behulp van deze instructie kunt u nu uw eigen speciale tekens of zelfs een geheel eigen karakterset maken. Als u in een programma zelfgemaakte tekens wilt gebruiken, moet u deze dus eerst zo definiëren voor u het hoofdprogramma start. Dat kan natuurlijk door dit programma aan zo'n hoofdprogramma vooraf te laten gaan. Omdat dit heel omslachtig is en veel geheugenplaatsen nodig heeft, zou u speciale programma's voor het maken van karaktersets in het geheugen kunnen laden, of met behulp van de zogenaamde overlay-techniek eerst het programma laden, dat de nieuwe karakterset maakt. Na afloop laadt dit programma zelf het hoofdprogramma in. Daarvoor moet u echter weten, hoe lang dit tweede BASIC-programma zal zijn. Als het korter is dan het programma van de tekendefinitie, heeft u geen problemen en kunt u door een

```
200 LOAD "programmanaam",8
```

bijvoorbeeld aan uw versie van ons voorbeeldprogrammaatje vastgeplakt het tweede programma invoeren, dat automatisch gestart wordt. Als het tweede programma langer dan het eerste is, moet u eerst door een:

```
190 POKE 45,AD - int(AD/256)*256
195 POKE 46,      int(AD/256)
```

meedelen, hoe lang het erna in te voeren programma zal zijn. Daarbij is AD het eindadres van het BASIC-geheugen. Dit kunt u in Simon's BASIC door de formule

AD = 30720 - FRE(O) + $801

berekenen als u het programma in het geheugen heeft. Dit van nu af aan vaste getal voegt u dan aan het eerste programma toe. Voor de geïnteresseerden onder u: 30720 is het aantal vrij ter beschikking staande bytes in Simon's BASIC als er geen programma in het geheugen zit; FRE(0) geeft het aantal vrij ter beschikking staande bytes, als uw programma in het geheugen staat. Het verschil 30720-FRE(0) berekent dus het aantal bytes, dat uw programma inneemt. Daar voegt u het begin-adres van het BASIC-programma (gewoonlijk $801) aan toe en dan heeft u al het eindadres van het BASIC-programma. U moet dus in het voorbeeldprogramma de regel

```
180 AD = 100
```

invoegen, als uw programma 100 bytes lang is. Tot zover een "ultrakorte" inleiding in de zogenaamde overlay-techniek, waarbij u natuurlijk op nog een paar dingen moet letten. U kunt meer BASIC-programma's achter elkaar uitvoeren. Als u niet alles begrepen hebt, is dat voor het begin verder niet erg. U moet echter absoluut een handboek aanschaffen, dat deze professionele programmering uitvoerig verklaart, zoals het boek "64 VOOR PROFS" van DATA-BECKER/Nederlands*.

```
100 REM ***************************
110 REM **                       **
120 REM **  @/DESIGN 2-VOORBEELD **
130 REM **                       **
140 REM ***************************
150 REM
160 MEM : REM KARAKTERSET VERSCHUIVEN
170 PRINT : REM LEGE REGEL
180 CENTRE "MISSCHIEN SCHRIJFT U AL UW VOLGENDE" : PRINT
190 CENTRE "TEKSTEN MET EEN ANDERE" : PRINT
200 CENTRE "KARAKTERSET" : PRINT
210 PRINT : REM LEGE REGEL
```

```
220 CENTRE "A B C D E"
230 PAUSE 10
240 REM
250 REM ************************
260 REM **                    **
270 REM **  TEKENDEFENITIE    **
280 REM **                    **
290 REM ************************
300 REM
310 DESIGN 2, $E000 + 1*8 : REM A
320 @ .BBBBBBB
330 @ ..BB..BB
340 @ .BBBBBBB
350 @ .BBBBBBB
360 @ ..BB..BB
370 @ ..BB..BB
380 @ .BBB..BB
390 @ ........
400 DESIGN 2, $E000 + 2*8 : REM B
410 @ .BBBBBBB
420 @ ..BB..BB
430 @ ..BB..BB
440 @ .BBBBBBB
450 @ ..BB..BB
460 @ ..BB..BB
470 @ .BBBBBBB
480 @ ........
490 DESIGN 2, $E000 + 3*8 : REM C
500 @ .BBBBBBB
510 @ ..BB..BB
520 @ ..BB....
530 @ ..BB....
540 @ ..BB....
550 @ ..BB..BB
560 @ .BBBBBBB
570 @ ........
580 DESIGN 2, $E000 + 4*8 : REM D
590 @ .BBBBBBB
600 @ ..BB..BB
610 @ ..BB..BB
620 @ ..BB..BB
630 @ ..BB..BB
640 @ ..BB..BB
650 @ .BBBBBBB
660 @ ........
670 DESIGN 2, $E000 + 5*8 : REM E
680 @ .BBBBBBB
690 @ ..BB..BB
700 @ ..BB....
```

```
710 @ ..BBBBB.
720 @ ..BB....
730 @ ..BB..BB
740 @ .BBBBBBB
750 @ ........
760 PAUSE 15
770 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
780 CENTRE "MET MEM KUNT U NATUURLIJK OOK ALLES":PRINT
790 CENTRE "WEER ONGEDAAN MAKEN"
800 PAUSE 10
810 MEM
820 PRINT : REM LEGE REGEL
830 CENTRE "ZIET U WEL?" : PRINT
840 PRINT : REM LEGE REGEL
850 CENTRE "A B C D E "
860 WAIT 198,255 : REM WACHTEN OP TOETSAANSLAG
```

13.4 CSET 0/1

Format	CSET 0 of CSET 1
Parameter	-------
Voorbeeld	CSET 0
Functie	Overschakelen op een tweede karakterset

CSET hebben we al in de vorm van CSET 2 in paragraaf 12.2.4 leren kennen. Daar had het de functie de grafiek in te schakelen, zonder haar uit te wissen. Nu leren we twee andere soorten van de CSET-instructie kennen. Hun functie is eigenlijk zeer gemakkelijk aan te geven. CSET 0 komt voor 100% overeen met de instructie NRM (paragraaf 9.3.3) en schakelt naast enkele andere dingen de hoofdletter/grafiek-modus in. Deze deelfunctie kunt u ook door

```
PRINT CHR$(142) ;
```

vervangen. Met CSET 1 nu heeft u de mogelijkheid om de alternatieve karakterset, dus het kleine letter-/hoofdletterschrift in te schakelen, terwijl de overige functies hetzelfde blijven (uitschakelen van de grafiek, van de Extended Colour-Modus en van de MEM-modus). De al genoemde deelfunctie is eveneens door

```
PRINT CHR$(14) ;
```

te bereiken. Voor een beter begrip nu een voorbeeld:

```
100 REM ************************
110 REM **                    **
120 REM **  CSET 0/1-VOORBEELD **
130 REM **                    **
140 REM ************************
150 REM
160 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
170 CSET 0 : REM HOOFDLETTERS/GRAFIEKTEKENS
180 CENTRE "OP HET OGENBLIK ZIJN WE IN DE" : PRINT
190 CENTRE "HOOFDLETTER / GRAFIEKTEKEN -MODUS" : PRINT
200 PRINT : REM LEGE REGEL
210 PAUSE 7
220 CENTRE "WAT DIREKT ZAL VERANDEREN !" : PRINT
230 PRINT : PRINT : REM 2 LEGE REGELS
240 PAUSE 5
250 CENTRE"LET OP!"
260 PAUSE 2
270 CSET 1 : REM KLEINE LETTERS / HOOFDLETTERS
280 WAIT 198,255 : REM WACHTEN OP TOETSAANSLAG
```

13.5 Testopdrachten

1. Waarop moet u letten bij het gebruik van MEM?

   A) MEM schakelt de grafiek uit
   B) MEM kopieert de karakterset na $EOOO
   C) Grafiek en eigen karakterset zijn niet tegelijkertijd mogelijk
   D) De MEM-situatie wordt door NRM en CSET 0 uitgeschakeld
   E) De MEM-situatie wordt door CSET 2 en CSET 1 uitgeschakeld

2. Waarop moet u letten bij DESIGN 2?

   A) DESIGN 2 schakelt de grafiek uit
   B) Met DESIGN 2 wordt een tekenpatroon vertaald

C) De parameter achter DESIGN 2 geeft het adres van de tekendefinitie aan
D) DESIGN 2 is met de grafiekvoorwaarden niet te gebruiken

Een van de meest in het oog springende mogelijkheden van uw Commodore 64 zijn natuurlijk de sprites of MOBs (= Movable Object Blocks = objecten, die bewegen). Sprites zijn zelfstandige grafiekjes die onafhankelijk van elkaar en van de rest van de beeldscherminhoud in het beeldschermvenster bewogen kunnen worden. In totaal heeft u de mogelijkheid om 8 sprites op hetzelfde moment op het beeldscherm te brengen. Sprites kunnen wat betreft hun kleur, hun grootte en de prioriteit ten opzichte van achtergrondtekens en elkaar gevarieerd worden. Ze kunnen raakvlakken tussen sprites onderling en met de achtergrond vaststellen. Al deze functies kunnen gemakkelijk met behulp van de VIC (videocontroller 6567) en diens registers gerealiseerd worden. Simon's BASIC vergemakkelijkt het gebruik van deze sprites enorm, zodat u alle functies overzichtelijk tot een paar instructies gereduceerd voor u ziet. Eerst zullen we ons even bezighouden met de opbouw van de sprites. Elke sprite bestaat uit 504 punten, die u afzonderlijk kunt plaatsen. Daarbij wordt een 24*21-puntenmatrix gebruikt, wat wil zeggen dat een sprite 24 punten breed en 21 punten hoog is. Binnen dit gebied kunt u figuren maken. We kunnen deze toedracht met een schetsje voorstellen:

```
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3

 0  . . . . . . . . . . . . . . . . . . . . . . . .
 1  . . . . . . . . . . . . . . . . . . . . . . . .
 2  . . . . . . . . . . . . . . . . . . . . . . . .
 3  . . . . . . . . . . . . . . . . .
 4  . . . . . . . .
enz.
20  . . . . . . . . . . . . . . . . . . . . . . . .
21  . . . . . . . . . . . . . . . . . . . . . . . .
```

We willen ons helemaal niet met de precieze opbouw van de sprites in het geheugen bezighouden, omdat dit in Simon's BASIC helemaal geen rol meer speelt. Hier kunnen sprites op een zeer eenvoudige manier gecreëerd worden. Binnen het voorgestelde gebied kunt u naar

hartelust uw figuur ontwerpen. U heeft de mogelijkheid om de kleur van de geplaatste punten apart van deze definitie uit 16 mogelijkheden te bepalen (zie beneden). Niet elke sprite echter heeft deze 24*21-opbouw. U kunt steeds tussen hoogoplossende en multicolour-sprites kiezen. De eerste hebben de zojuist beschreven opbouw. De multicolour-sprite daarentegen wordt net als de multicolour-grafiek gemaakt. Daarom heeft zo'n sprite in x-richting de halve oplossing. Hier zijn dus slechts 12, maar twee keer zo brede punten in een regel. Daarbij kan een sprite uit in totaal vier verschillende kleuren (met de achtergrondkleur) samengesteld zijn, terwijl een hoogoplossende (HGR) sprite slechts twee kleuren heeft. Deze vier kleuren zijn in verschillende registers van de VIC ondergebracht (zie CBM 64-gebruikershandboek pagina 153/154). Onder voorbehoud moet echter nog gezegd worden, dat alleen de kleur 2 voor alle sprites verschillend kan zijn. De andere kleuren (kleur 1 en 3 en de achtergrondkleur) zijn telkens voor alle sprites gelijk, omdat ze uit dezelfde registers geput worden. In tegenspraak met de informatie in het CBM 64 gebruikershandboek kunt u ook in multicolour alle 16 kleuren voor uw figuren gebruiken.

Naast de kleur en de definitie van de sprites kunnen er nog enkele dingen veranderd worden. U heeft de mogelijkheid om uw produkt in x- of in y-richting of in beide richtingen met de factor 2 te vergroten. Dat wil zeggen dat uw sprite dubbel zo breed, dubbel zo hoog of beide wordt. Bovendien geeft u aan, welke prioriteit de sprite op dat moment voor de achtergrond heeft. Met andere woorden: of uw sprite voor de tekens of de rest van de grafiek staat (of beweegt), of dat hij zich erachter bevindt en wordt afgedekt door de rest van de scherminhoud. Tegelijk bepaalt u door het nummer van een sprite (elk van de 8 mogelijke sprites heeft een nummer van 0 tot 7) de prioriteit van de sprites onderling. De sprites met de hogere nummers hebben de hoogste prioriteit en bedekken bij overlappingen dus steeds de sprites met een lager nummer. Zo kunt u dus op een eenvoudige manier 3-dimensionale effecten produceren.

Maar dat is nog niet alles. Uw computer registreert namelijk elke aanraking van een sprite met een achtergrondteken (of met een gezette punt op het beeldscherm) of met een andere sprite. Deze gebeurtenis wordt eveneens in de verschillende register van de VIC vastgelegd, wat verder niet voor u van belang is, omdat Simon's BASIC dit werk voor u doet. Hierbij zijn slechts een paar dingen van belang, die onder de instructies van dit moment (DETECT en CHECK) behandeld worden. Belangrijk is alleen het feit, dat tussen aanrakingen achtergrond-sprite en sprite-sprite verschil kan worden gemaakt. In het laatste geval kan er nog uitgemaakt worden, welke van de 8 sprites elkaar aangeraakt hebben. Een ding moet hier nog even verklaard worden: het opslaan van een sprite (MOBs) in het geheugen van uw CBM 64. Bij het opslaan van een sprite wordt het totale schrijf/leesgeheugen van de computer in zogenaamde blokken van elk

64 bytes verdeeld. Elk blok krijgt een specifiek nummer. Op die manier bezitten de eerste 64 bytes van het geheugen (adressen 0-63) het bloknummer 0 etc. Als u nu een spritedefinitie in het geheugen wilt plaatsen (wat met de instructie DESIGN gerealiseerd wordt), moet u deze eerst even binnen zo'n blok plaatsen. Later geeft u dan bij de afbeelding van een sprite aan, uit welk blok de computer de definitie (of het tekenpatroon) moet halen. Op deze manier is het bijvoorbeeld mogelijk om twee sprites er identiek uit te laten zien doordat u twee verschillende spritenummers in hetzelfde geheugenblok opslaat. Bij de nummering van een blok bent u beperkt tot de waarden 0-255. Daarom kunt u uw verschillende figuren slechts binnen 255*64=16384 bytes (16 K) onderbrengen. Waar deze 16 K in het geheugen van uw computer liggen, bepaalt de situatie van uw videoram (beeldschermgeheugen). Deze is in Simon's BASIC als volgt bepaald:

Toestand	Blokpositie
normale tekstmodus:	0 - 16383 ($0000 - $3FFF)
nieuwe karakterset:	49152 - 65536 ($C000 - $FFFF)
grafiekmodus :	49152 - 65536 ($C000 - $FFFF)

Als u tegelijkertijd in grafiek en in tekst werkt, liggen de spritedefinities op verschillende geheugenplaatsen, hoewel de bloknummers identiek zijn. Als u dus afwisselend in grafiek en tekst werkt, moet u met dit gegeven rekening houden: een sprite, die in de tekstmodus in het geheugen opgeslagen werd, zal in de grafiek niet of slechts met een volkomen onzinnig uiterlijk opduiken. Dat gebeurt, omdat het geheugenbereik, dat voor het betreffende blok betrekking heeft, de spritedefinitie natuurlijk niet bevat. Omgekeerd geldt hetzelfde. Dat kunt u alleen vermijden, doordat u dezelfde sprite een tweede keer in het grafiek-blok-bereik opslaat, of hem bij de omschakeling op grafiek uitschakelt. Nog op een element moet u bij de keuze van het blok voor een sprite-definitie letten: vermijd u de blokken, die geheugengebieden kenmerken, die al door andere elementen bezet zijn (bijv. zeropage). Hieronder worden een paar mogelijke blokken opgesteld:

Tekstmodus

bloknummer	adressen	opmerkingen
13-15	832-1023 ($0340-$03FF)	zonder beperking te gebruiken
64-255	4096-16383 ($1000-$3FFF)	voorzichtig bij lange BASIC-programma's

bloknummer	adressen	opmerkingen
16-23	50176-53247 ($C400-$CFFF)	met IEC-bus van Data becker (S versie) slechts tot blok 47

nieuwe karakterset:

bloknummer	adressen	opmerkingen
16-47	50176-52223 ($C400-$CBFF)	beperkt te gebruiken
192-255	61440-65535 ($F000-$FFFF)	onbeperkt te gebruiken

Bij het gebruik van de C-pagina, blokken 16-63 in de grafiekmodus en bij gebruik van een nieuwe karakterset, moet u oppassen. Ergens in dit gebied liggen een paar tussengeheugens van Simon's BASIC, o.a. de reservering van de functietoetsen (blokken 25-30). Het zou mogelijk zijn, dat er in dit gebied nog andere tussengeheugens liggen, die voor het uitvallen van bepaalde functies verantwoordelijk zouden kunnen zijn. Als u dus met dit gebied werkt, moet u eerst even uw programma opslaan, voordat u met de eerste proefstart begint. Dat moet u trouwens altijd doen, omdat Simon's BASIC af en toe neigingen heeft om in te storten.

U ziet dus, dat u met de sprites mogelijkheden aan de hand gedaan worden, die u pas bij het proberen en het gebruik volledig kunt benutten. De afzonderlijke voorbeeldprogramma's na de verklaring van de verschillende instructies moeten u daaromtrent aanwijzingen geven, die u bij uw volgende programma misschien kunt gebruiken. Allereerst worden die instructies verklaard, die voor het begin van een spritedefinitie nodig zijn, voordat u deze op het beeldscherm te zien krijgt. Uiteraard kunnen deze zaken in de loop van het programma weer veranderd worden, maar ze moeten toch voordat ze zichtbaar worden, gedefinieerd zijn. Daarna wordt u met instructies geconfronteerd, die de kant en klare sprites uiteindelijk besturen. Laat u zich maar verrassen. Veel plezier!

14.1 Spritedefinities

14.1.1 DESIGN 0/1

Format	DESIGN 0, ba + b*64 of DESIGN 1, ba + b*64
Parameter	0 : hoogoplossende sprite 1 : multicolour-sprite ba: basisadres van het videobereik: tekstmodus : ba=0 grafiek : ba=$COOO nieuwe karakterset : ba=$COOO b : nummer van het geheugenblok, waarnaar de spritedefinitie overgeplaatst moet worden (0-255)
Voorbeeld	DESIGN 0,$COOO + 13 * 64
Functie	Overplaatsen van een spritedefinitie naar een geheugenbereik

U zult zich ongetwijfeld nog de instructie DESIGN 2 uit paragraaf 13.2 kunnen herinneren, die een rol speelde bij de definitie van nieuwe tekens. Als u dit gedeelte van het boek nog niet hebt gelezen, zou u dat eerst moeten doen. DESIGN 0 en DESIGN 1 werken net zo als de in dat hoofdstuk besproken instructie. Met DESIGN 0/1 plaatst u de definitie van een sprite (MOBs), die (zoals ook bij de verandering van de karakterset) in bitpatroonregels (apestaartregels: zie 14.1.2) opgeslagen wordt, naar het daarvoor verantwoordelijke geheugen over. De instructie DESIGN (0, 1, of 2) moet steeds direkt in de regel voor de bitpatroonregels staan, om de foutmelding BAD CHAR FOR A MOB te vermijden. De waarden 1 of 0 bepalen welke soort spritedefinitie ze in het geheugen willen plaatsen. Daarbij heeft u de mogelijkheid om tussen hoogoplossende (HGR-)sprites met slechts een puntkleur (naast de achtergrondkleur) en multicolour-sprites te kiezen, waarbij maximaal drie kleuren (met inachtneming van de genoemde beperkingen) voor de presentatie van een figuur gebruikt kunnen worden. Als u nu de definitie van een HGR-sprite naar het geheugen wilt overplaatsen, moet u het cijfer 0 meteen achter het DESIGN-woord (DESIGN 0) kiezen. In het geval van een multocolour-sprite moet dat dus zijn: DESIGN 1. De tweede en derde parameter dienen voor de bepaling van het geheugenbereik, waarnaar de definitie overgeplaatst moet worden. Daarbij moet u wat in de

inleiding is gezegd ter harte nemen. ba geeft hier dus het adres van blok 0 (basisadres van het video-adresbereik) en b het nummer van het desbetreffende blok aan. Alle overige informatie kunt u vinden in de inleiding van hoofdstuk 14 of in paragraaf 14.1.2.

14.1.2 Apestaart @

Format	apestaart ...bitpatroonregel...
Parameters	bitpatroonregel: spritedefinitie
	punt (.) : lege positie
	B : punt met de kleur 1
	C : punt met de kleur 2
	D : punt met de kleur 3
Voorbeeld	apestaart ...B..BBBB....B...B.B.B.
	apestaart ... etc. (21 regels)
Functie	Maken van een spritedefinitie

Ook deze instructie kent u al uit hoofdstuk 13 als kenmerk van een bitpatroonregel van een teken in tekstmodus. Hier echter dient de apestaart ertoe om een patroonregel in te leiden, die de definitie van een regel van een sprite bevat. Hoe sprites opgebouwd zijn, zou u uit uw gebruikershandboek of uit de inleiding van dit hoofdstuk bekend moeten zijn. Er wordt een verschil gemaakt tussen HGR- en multicolour-sprites, die dus eveneens verschillend gedefinieerd moeten worden. Bij het gebruik van hoogoplossende (HGR-) sprites heeft u een 24*21-puntenmatrix, waarin aangegeven wordt of een punt wel of niet geplaatst is. In Simon's BASIC is een HGR-sprite dus in totaal uit 21 patroonregels met elk 24 puntaanwijzingen samengesteld. Een geplaatste punt wordt door een B in deze definitie voorgesteld; een niet-geplaatste punt, die de achtergrondkleur aanneemt, door een gewone punt. In multicolour heeft u naast de kleur, die u voor elke sprite extra kunt kiezen, nog twee andere kleuren, die alle sprites gemeenschappelijk hebben. Daarom staan hiervoor de letters C en D ter beschikking om de betreffende kleuren voor een punt te symboliseren. Tegelijkertijd kunnen er nog maar 12 puntdefinities in

een regel vorkomen, want het oplossend vermogen van multicolour is slechts half zo groot. De vorm van de definitie kunt u ontlenen aan het voorbeeldprogramma.

Er is nog iets waar u op moet letten bij het gebruik van de letters: in HGR nemen de sprites de kleur aan van de met B gemerkte punten via de MOB SET-instructie. Bij multicolour sprites wordt de kleur van de letter C aangenomen. Zo krijgen we de volgende toewijzingen:

Hoogoplossende sprites:

letter	kleurnummer	passende kleurinstructie
.	0	(transparant)
B	1	MOB SET

Multicolour sprites:

letter	kleurnummer	passende kleurinstructie
.	0	(transparant)
B	1	CMOB
C	2	MOB SET
D	3	CMOB

Zou u op het idee komen om behalve spaties onder bepaalde voorwaarden ook andere lettertekens in de spritedefinitie op te nemen, de regels langer te maken dan is toegestaan of tussen de instructie DESIGN en de bitpatroonregels een BASIC-regel in te voegen, dan moeten we u helaas waarschuwen dat uw computer onder die omstandigheden niet bereid is met u samen te werken. Hij meldt zijn ontstemming dan met BAD CHAR FOR A MOB. Ook computers kunnen het moment bereiken dat ze zich schaamteloos misbruikt voelen. Zorgt u er dus voor dat zich in wederzijds belang dergelijke situaties niet voordoen. Uw computer zal uw zorgvuldigheid beantwoorden met een hechte vriendschap.

Voorbeelden:

```
100 REM ************************
110 REM **                    **
120 REM **  @/DESIGN-VOORBEELD **
130 REM **                    **
140 REM ************************
150 REM
```

```
160 DESIGN 1,14*64 :REM DEFINITIE IN BLOK 14 OVERNEMEN
170 @...BBBBB....
180 @ .BBB...BBB..
190 @ BB.CC.CC.BB.
200 @ BB...C...BB.
210 @ BB...C...BB.
220 @ .BBBCCCBBB..
230 @ ...BBBBB....
240 @ B...DDD.....
250 @ BB..DDD.....
260 @ BBBBDDDBB...
270 @ ..BBDDDBB...
280 @ ....DDD.BB..
290 @ ....DDD..BB.
300 @ ....DDD...B.
310 @ ....DDD.....
320 @ ..CCC.CCC...
330 @ .CCC...CCC..
340 @ .CC.....CC..
350 @ CCC.....CCC.
360 @ CCC.....CCC.
370 @ BBBBBBBBBBBB
380 REM
390 REM *************************
400 REM *****          **********
410 REM ***** AANVULLING **********
420 REM *****          **********
430 REM *************************
440 REM
450 COLOUR 9,9
460 MOB SET 1,13,8,0,1
470 CMOB 7,5
480 MMOB 1,0,0,300,205,2,200
```

tussen de @ en de .. BC etc mag géén spatie staan

14.1.3 MOB SET

Format MOB SET n, b, k, p, o

Parameter n: nummer van de sprite (0-7)

b: nummer van het blok (0-255) waaruit de spritedefinitie wordt opgehaald en wordt toegekend aan het spritenummer

k: kleur van de sprite (0-15) HGR: de letter B / MC : de letter C

p: prioriteit van de sprite ten opzichte van achtergrondtekens (p=0 : sprite voor achtergrond / p=1 : sprite na achtergrond)

o: oplossing van de sprite (o=0 : HGR-sprite / o=1 : MC-sprite)

Voorbeeld MOB SET 0,13,7,1,0

Functie eigenschappen van een sprite bepalen

U kunt verschillende eigenschappen van sprites afzonderlijk specificeren. Dat kan onafhankelijk van een voorafgaande definitie. Een deel van deze functies wordt vastgelegd met de instructie MOB SET, waarbij u een flinke hoeveelheid parameters kunt specificeren die elk een eigen betekenis hebben. Deze instructie moet u zien als een aanvulling op uw spritedefinitie, want er wordt niet een functie mee bereikt als het aan- of uitschakelen van een sprite, of het bewegen daarvan op een beeldscherm.

Nu over naar de bespreking van de afzonderlijke parameters. Ten eerste geeft u met n aan om welke sprite het eigenlijk gaat. U kunt maximaal 8 sprites tegelijkertijd op het beeldscherm presenteren. Elke sprite heeft daarbij een heel specifiek nummer (0-7), dat overigens ook de prioriteit van de afzonderlijke sprites onder elkaar regelt (zie inleiding). Dit nummer moet u bij alle sprite-instructies aangeven, om te bepalen op welke sprite die instructie betrekking heeft. De tweede parameter b bepaalt het blok, waaruit de patroondefinitie moet komen, die van tevoren met de intussen bekende instructies DESIGN en APESTAART uit het geheugen overgeplaatst werd. Deze definitie wordt nu in onze MOB SET-instructie met het aangegeven spritenummer samengevoegd, zodat later deze ene figuur steeds door dit nummer opgeroepen wordt. Op deze manier is het bijvoorbeeld eveneens mogelijk om meer identieke sprites op het beeldscherm te presenteren, doordat aan hetzelfde blok verschillende spritenummers toegevoegd worden. Bij de keuze van het blok moet u op de Dit voorbeeld dient speciaal om u vertrouwd te maken met het definiëren van sprites met behulp van de twee bekende instructies. Er is echter nog een reeks andere instructies nodig om ze op het beeldscherm te brengen. Die hebben we eerst maar even zonder commentaar vanaf regel 460 aan het programma toegevoegd. Zo kunt u toch uw produkt in natura bekijken. We komen later natuurlijk nog op deze instructies terug.
elementen letten, die in de inleiding beschreven zijn.

Om de kleur van een sprite te bepalen, is het noodzakelijk om aan de

volgende parameter k een waarde te geven. Deze waarde stelt de kleurcode (0-15) voor en heeft in HGR- en multicolour-sprites een verschillende betekenis: in hoogoplossende figuren worden alle punten van de in de patroonregels aangegeven matrix, die met een B gemerkt werden, in de aangegeven kleur getekend. In multicolour-sprites daarentegen hebben alle met C getekende punten deze taak. De volgende waarde legt de prioriteit van het object voor de overige achtergrondtekens van het beeldscherm vast. p kan slechts twee waarden aannemen: 0 of 1. In het eerste geval beweegt de figuur steeds voor de overige tekens of punten (ze maakt de achtergrond dus onzichtbaar). In het tweede geval voor p=1 wordt omgekeerd de sprite onzichtbaar. Tenslotte legt u vast, of de spritedefinitie in hoogoplossende (o=0) of multicolourgrafiek (o=1) gepresenteerd moet worden. U kunt dus ook een au fond voor multicolour bedoelde definitie onder HGR (en ongekeerd) laten lopen. Nu maar een voorbeeld:

```
100 REM ************************
110 REM **                    **
120 REM **  MOB SET-VOORBEELD **
130 REM **                    **
140 REM ************************
150 REM
160 COLOUR 0,0 : REM VENSTER- ACHTERGRONDKLEUR = ZWART
170 DESIGN 0,13*64 : SPRITEDEFINIERING IN BLOK 13
190 @ ...........BB...........
200 @ ..........BBBB..........
210 @ ..........BBBB..........
220 @ ...........BB...........
230 @ .....BB....BB....BB.....
240 @ ......BB...BB...BB......
250 @ .......BB..BB..BB.......
260 @ ........BB.BB.BB........
270 @ ..BB.....BBBBBB......BB.
280 @ .BBBBBBBBBBBBBBBBBBBBBBB
290 @ .BBBBBBBBBBBBBBBBBBBBBBB
300 @ ..BB.....BBBBBB......BB.
310 @ ........BB.BB.BB........
320 @ .......BB..BB..BB.......
330 @ ......BB...BB...BB......
340 @ .....BB....BB....BB.....
350 @ ...........BB...........
360 @ ..........BBBB..........
370 @ ..........BBBB..........
380 @ ...........BB...........
390 @ ........................
400 REM
410 MMOB 0,100,100,100,100,3,0 : REM SPRITE OP HET BEELDSCHERM
    BRENGEN
```

```
420 FOR X=0 TO 15
430 MOB SET  0,13,X,0,0 : REM EIGENSCHAPEN BEPALEN EN KLEUR WISSELEN
440 PAUSE 2
450 NEXT X
460 COLOUR 8,8
470 REM
480 FOR X=1 TO 6
490 MOB SET 0,13,7,1,0 : REM ACHTER DE BEELDSCHERMTEKENS
500 PAUSE 1
510 MOB SET  0,13,7,0,0 : REM VOOR DE BEELDSCHERMTEKENS
520 PAUSE 1
530 NEXT X
```

Ook hier wordt met de instructie MMOB op het nog onbekende vooruitgelopen, om de sprite tenminste zichtbaar te maken. Stoort u zich er verder maar niet aan.

14.1.4 CMOB

Format	CMOB k1, k3
Parameter	k1 : kleurnummer 1 voor multicoloursprites (letter B) (0-15)
	k3 : kleurnummer 3 voor multicoloursprites (letter D) (0-15)
Voorbeeld	CMOB 2,7
Functie	Kiezen van extra kleuren voor multicoloursprites

De instructie stelt een aanvulling voor van de MOB SET-instructie, speciaal voor multicoloursprites. Er kunnen in een multicoloursprite 3 puntkleuren bij het maken van een figuur een rol spelen (1 kleur in HGR-sprites). Deze 2 extra kleuren kunt u met behulp van de CMOB-instructie vastleggen. Daarbij staat k1 voor de kleur, die u bij de definitie van de minigrafiek in de bitpatroonregels met een B symboliseert. Alle punten, die u met een B gemerkt hebt, worden nu in kleurnummer 3 (k3) op het beeldscherm gepresenteerd. Het opgeven van de kleur van de instructie SET MOB legt kleur 2 van de sprites, dus alle met C getekende punten van de spritedefinitie, vast. Belangrijk in dit verband is verder nog, dat de twee met CMOB

uitgekozen kleuren voor alle (max. 8) sprites tegelijkertijd gelden, wat betekent dat alle sprites met deze kleuren identiek zijn. Alleen de met C gemerkte punten kunnen voor elke sprite apart gekozen worden. Voor het gebruik van deze instructie staan u in de verschillende paragrafen diverse voorbeelden ter beschikking.

14.1.5 Testopdrachten

1. Welke syntax is juist?

   A) DESIGN 0
   B) MOB SET 1,15,3,0,0
   C) DESIGN 0,832
   D) CMOB 2,3

2. Waar moet u op letten bij het gebruik van DESIGN?

   A) DESIGN kan men slechts voor de afbeelding van een sprite gebruiken.
   B) Het blok 13 beslaat in de grafiekmodus een ander geheugenbereik dan in de tekstmodus.
   C) De eerste parameter geeft aan, of het om een kleine of een vergrote sprite gaat.
   D) Tussen DESIGN en de eerste patroonregel mag geen andere BASIC-regel staan.
   E) DESIGN sluit multicoloursprites uit.

3. Wat voor effect hebben de eerste twee parameters van de MOB SET-instructie?

   A) Ze geven aan of het om een multicolour- of een HGR-sprite gaat en bepalen de prioriteit.
   B) De definitie uit het genoemde blok wordt met behulp van het aangegeven nummer met de sprite samengevoegd.
   C) De eerste parameter beeldt de sprite op het beeldscherm af.
   D) De tweede parameter geeft aan, welk uiterlijk de sprite moet hebben.

14.2 Spritebesturing

14.2.1 MMOB

Format	MMOB n, x1, y1, x2, y2, gr, ge
Parameter	n : nummer van de te bewegen sprite (0-7)
	x1: x-coördinaat van het beginpunt (0-511)
	y1: y-coördinaat van het beginpunt (0-255)
	x2: x-coördinaat van het bestemmingspunt (0-511)
	y2: y-coördinaat van het bestemmingspunt (0-255)
	gr: grootte van de sprite (0-3)
	ge: snelheid, waarmee de sprite zich tussen de twee punten moet bewegen (0-255)
Voorbeeld	MMOB 1,100,50,200,160,0,10
Functie	Presenteren en bewegen van een sprite op het beeldscherm

MMOB maakt ons een paar dingen mogelijk: ten eerste schakelt het een sprite in, zodat deze op het beeldscherm te zien is. Maar tegelijkertijd worden er nog enkele andere functies mogelijk gemaakt. U kunt de positie aangeven, waarop uw figuur voor de eerste keer op het beeldscherm verschijnt. Deze positie geeft u met x1 en y1 als coördinaten aan. Ten tweede kunt u twee andere coördinaten x2 en y2 benoemen, die bepalen naar welk punt de sprite, dadelijk nadat hij opgedoken is, moet gaan. Eerst iets over de coördinaten: principieel komen de coördinaten, die u als spritecoördinaten opgeeft, niet met de gebruikelijke beeldschermgrafiek-coördinaten overeen. Er worden steeds coördinaten opgegeven, waar de linker benedenhoek van de sprite zich bevindt. Als dit punt zich in het ons bekende nulpunt van de grafiek bevindt (linker bovenhoek van het beeldschermvenster), dan zijn de spritecoördinaten daar al x=20 en y=30. Zo geldt deze nulpuntverschuiving voor het totale beeldscherm. Verder kunt u ver over het venster heen waarden opgeven (x: 0-511 ; y: 0-255). Dit moet kunnen om een sprite ook uit het beeldvenster te laten bewegen. Als

u de sprite enkel en alleen op een bepaalde plaats op het beeldscherm wilt neerzetten, zonder hem te laten bewegen, kiest u de beide coördinatenparen x1,y1 en x2,y2 identiek. Nu is het beginpunt gelijk aan het bestemmingspunt en uw sprite blijft staan. Uw sprite zal zich van het beginpunt (x1,y1) tot het bestemmingspunt (x2,y2) dwars over de tekst of de grafiek heen bewegen. Dat gebeurt met de snelheid, die u als laatste parameter ge aangeeft. ge kan waarden van 0-255 aannemen. Hoe kleiner ge gemaakt wordt, hoe sneller de sprite wordt. Bij ge=255 dus, tuft uw slang op z'n gemak door de bosjes, terwijl de bij ge=0 vliegende starfighter over uw hoofd wegraast. Maar wacht even, we hebben nog iets vergeten. Hoewel u het niet bij deze instructie zou verwachten: met de parameter gr kunt u tegelijkertijd bij de uitvoering van deze instructie de grootte van de figuur bepalen. U kunt elke sprite in twee richtingen vergroten. De mogelijkheden, die gr u verschaft, worden in een kleine tabel weergegeven:

gr	Vergroting	vergrotingsfactor	puntmatrix
0	geen	1x1	24x21
1	x-richting	2x1	48x21
2	y-richting	1x2	24x42
3	x/y-richting	2x2	48x42

Met puntmatrix wordt hier natuurlijk die matrix bedoeld, die op het beeldscherm verschijnt, dus het aantal punten van het beeldscherm dat door een sprite maximaal bedekt wordt. Om het geheel iets duidelijker te maken, wordt hier weer een voorbeeldje gegeven:

```
100 REM ********************
110 REM **                **
120 REM **  MMOB-VOORBEELD **
130 REM **                **
140 REM ********************
150 REM
160 COLOUR 0,0 : REM VENSTER- ACHTERGRONDKLEUR = ZWART
170 DESIGN 0,13*64 : SPRITEDEFINITIE IN BLOK 13
180 @ ........................
190 @ ........................
200 @ ........................
210 @ ........................
220 @ ........................
230 @ ........................
240 @ ........................
250 @ ...BB...................
```

```
260 @ ...BBB..................
270 @ ...BBBB.................
280 @ ...BBBBB.......BBBB.....
290 @ ...BBBBBB.....BBB.BB....
300 @ ...BBBBBBBBBBBBBBBBB...
310 @ .BBBBB.BB.BB.BB.BB.BBBBB
320 @ ...BBBBBBBBBBBBBBBB....
330 @ ........................
340 @ ........................
350 @ ........................
360 @ ........................
370 @ ........................
380 @ ........................
390 REM
400 MOB SET 0,13,7,0,0 : REM SPRITE 0 = GEEL
410 MOB SET 1,13,2,0,0 : REM SPRITE 0 = ROOD
420 REM
430 REM ***************
440 REM **           **
450 REM **  BEWEGING **
460 REM **           **
470 REM ***************
480 REM
490 MMOB 0,0,0,200,200,1,50 : REM IN X-RICHTING UITGEREKT
500 MMOB 1,0,100,220,220,0,70 : REM NORMAAL
510 PAUSE 5
520 COLOUR 9,9 : REM VENSTER- ACHTERGRONDKLEUR = BRUIN
```

Zoals u ziet, worden er hier twee sprites, die er identiek uitzien, maar verschillend in kleur en grootte zijn, snel over het beeldscherm heen bewogen. Let u op de regels 400 en 410, waarin sprite 0 en 1 hetzelfde blok toegewezen wordt.

14.2.2 MOB OFF

Format	MOB OFF n
Parameter	n : nummer van de uit te schakelen sprite (0-7)
Voorbeeld	MOB OFF 0
Functie	Uitschakelen van een sprite

Nadat u geleerd heeft hoe u een sprite op het beeldscherm kunt krijgen (MMOB), moeten we ons natuurlijk ook met de instructie bezighouden, die het ons mogelijk maakt dit object weer uit te wissen. Om deze opgave te vervullen, staat de instructie MOB OFF klaar. U geeft als enige parameter het nummer van de uit te schakelen sprite aan en u ziet al niets meer op uw scherm. Maar MOB OFF wist alle sprite-toevoegingen bij bepaalde definitieblokken uit (de inhoud van de blokken blijft natuurlijk behouden en kan ook verder gebruikt worden). Dat wil zeggen: u moet, als u deze sprite weer inschakelt, dus weer op het beeldscherm wilt presenteren, het desbetreffende spritenummer weer door middel van MOB SET een bepaald blok met een spritedefinitie toewijzen. De met CMOB gekozen twee extra multicolourkleuren blijven daarentegen behouden. In aansluiting hierop wordt het object natuurlijk weer met MMOB op het beeldscherm te voorschijn getoverd. Met deze instructie is het mogelijk bijvoorbeeld getroffen ruimteschepen te elimineren, of een beweegbaar figuur te presenteren, doordat u tussen twee of meer spritedefinities heen en weer schakelt. Een tip: u kunt MOB OFF uiteraard ook vervangen door de betreffende sprite uit het beeldscherm weg te positioneren (met MMOB), zodat hij evenmin nog te zien is. Het voordeel daarbij: de bloktoevoeging blijft behouden. Dat betekent, dat u uw sprite op een willekeurig moment weer kunt laten opduiken, zonder uitgebreid de instructie MOB SET in te hoeven typen. Als u daarentegen de definitie wilt verwisselen, dus de sprite wilt veranderen, hoeft u voor de MOB SET-instructie niet absoluut een MOB OFF in te voeren.

Voorbeeld:

```
100 REM ***********************
110 REM **                   **
120 REM **  MOB OFF-VOORBEELD **
130 REM **                   **
140 REM ***********************
150 REM
160 COLOUR 0,0 : REM VENSTER- ACHTERGRONDKLEUR = ZWART
170 DESIGN 0,13*64 : REM SPRITEDEFINITIE IN BLOK 13
180 @ ........................
190 @ ........................
200 @ ..........B..B..........
210 @ ..........B..B..........
220 @ ..........B..B..........
230 @ ..........B..B..........
240 @ ..........BBBB..........
250 @ .........BBBBBB.........
260 @ ........BB....BB........
270 @ ...B...BB...BB.BB...B...
280 @ ...B..BB..B.BB..BB..B...
```

```
290 @ ...BBBBBBBBBBBBBBBBBB...
300 @ ..BBBBB.B.B.B.B.B.BBBB..
310 @ .BBBBB.B.B.B.B.B.BBBBBB.
320 @ BBBBBBBBBBBBBBBBBBBBBBBB
330 @ BB.....BBBBBBBBBB.....BB
340 @ BB.....BB......BB.....BB
350 @ ......BB........BB......
360 @ ....BBBB........BBBB....
370 @ ....BBBB........BBBB....
380 @ ........................
390 REM
400 REM
410 DESIGN 0,14*64 : REM SPRITEDEFINITIE IN BLOK 14
420 @ ........................
430 @ ........................
440 @ ......BB........BB......
450 @ .......BB......BB.......
460 @ ........BB....BB........
470 @ .........BB..BB.........
480 @ ..........BBBB..........
490 @ .........BBBBBB.........
500 @ ........BB....BB........
510 @ .......BB...BB.BB.......
520 @ ......BB..B.BB..BB......
530 @ ...BBBBBBBBBBBBBBBBBB...
540 @ ..BBBB.B.B.B.B.B.BBBBB..
550 @ .BBBBBB.B.B.B.B.BBBBBBB.
560 @ BBBBBBBBBBBBBBBBBBBBBBBB
570 @ .......BBBBBBBBBB.......
580 @ .......BB..BB..BB.......
590 @ ......BB........BB......
600 @ ....BBBB........BBBB....
610 @ ....BBBB........BBBB....
620 @ ........................
630 REM
640 REM
650 REM ***************
660 REM **           **
670 REM **  FLITSEN  **
680 REM **           **
690 REM ***************
700 REM
710 FOR X=1 TO 100
720 REM ---------------
730 MOB SET 0,13,6,0,0
740 MMOB 0,200,100,200,100,3,0 : REM EERSTE SPRITE  INSCHAKELEN
750 FOR Y=1 TO 60 : NEXT Y
760 MOB OFF 0 : REM TWEEDE SPRITE UITSCHAKELEN
770 REM ---------------
```

```
780 MOB SET 0,14,6,0,0
790 MMOB 0,200,100,200,100,3,0 : REM TWEEDE SPRITE  INSCHAKELEN
800 FOR Y=1 TO 60 : NEXT Y
810 MOB OFF 0 : REM TWEEDE SPRITE UITSCHAKELEN
820 REM ---------------
830 NEXT X
840 COLOUR 9,9
```

In dit voorbeeld wordt door de constante wisseling tussen twee soortgelijke spritedefinities een oplichten of bewegen gesimuleerd. De regels 760 en 810 schakelen de oude sprite telkens uit en kunnen eigenlijk ook weggelaten worden. Zo is het mogelijk beweegbare figuren te maken. Als deze figuur bovendien nog uit meer sprites is samengesteld, krijgt u echt leuke effecten. Probeert u er maar iets mee te doen, maar kijkt u ook even naar de volgende instructie en het daarbijbehorende voorbeeld. U zult zich verbazen!

14.2.3 RLOCMOB

Format	RLOCMOB n, x2, y2, gr, ge
Parameter	n : nummer van de te bewegen sprite (0-7)
	x2: x-coördinaat van het bestemmingspunt (0-511)
	y2: y-coördinaat van het bestemmingspunt (0-255)
	gr: grootte van de sprite (0-3)
	ge: snelheid, waarmee de sprite zich naar het punt moet bewegen (0-255)
Voorbeeld	RLOCMOB 1,200,160,0,10
Functie	Presenteren of verder bewegen van een sprite op het beeldscherm

U krijgt nog een andere, zeer nuttige instructie, die de mogelijkheid heeft een sprite vanuit zijn oorspronkelijke (bijvoorbeeld met MMOB ingestelde) positie naar een tweede bestemmingspunt te verplaatsen. Daardoor hoeft u niet - zoals in de MMOB-instructie- voortdurend ook het beginpunt aan te geven en dit te onthouden. Verder functioneert

deze instructie voor 100% analoog aan MMOB. U moet dus de uiteenzettingen bij deze in paragraaf 14.2.1 behandelde instructie goed gelezen en begrepen hebben. RLOCMOB schakelt dus eveneens een besproken sprite aan. Met n bepaalt u het spritenummer, gr definieert de grootte en ge de snelheid waarmee hij zich moet voortbewegen. Gebruik RLOCMOB liever niet om een sprite in te schakelen, omdat hij anders dwars over het beeldscherm heen naar de aangegeven positie vliegt, omdat de oorsprongcoördinaat natuurlijk nog niet bepaald is. En dit is hoogstwaarschijnlijk niet in uw belang. Met deze instructie wordt het zeer eenvoudig gecontroleerde bewegingen van meer sprites gelijktijdig toe te passen. Het Simon's BASIC-handboek geeft u hiervan een heel mooi voorbeeld op de pagina's 51-53. We zullen ons nu op de praktijk richten met een voorbeeld.

```
100 REM ************************
110 REM **                    **
120 REM **  RLOCMOB-VOORBEELD **
130 REM **                    **
140 REM ************************
150 REM
160 COLOUR 0,1 : REM VENSTER- ACHTERGRONDKLEUR = ZWART
170 DESIGN 0,13*64 : REM SPRITEDEFINITIE IN BLOK 13
180 @ .............BBB........
190 @ ..BBB.......BB.BB.......
200 @ ...BB..B...BBBBBBBB.....
210 @ .BBBBBB....BBBBBBB..BB..
220 @ ...BB..B....BB..BBBBBB..
230 @ ..BBB........BBB........
240 @ ............BBB.........
250 @ ............BBB.........
260 @ ............BBBBB.......
270 @ ............BBB.BB......
280 @ ...........BBBB..BB.....
290 @ ..........BBBBB...BB....
300 @ ..........B.BBB.........
310 @ .........B..BBBB........
320 @ ........B..BB.BBB.......
330 @ .......B...BB..BBB......
340 @ ......B....BB....BB.....
350 @ ......BB...BB....BB.....
360 @ .BBBBBBB...BB....BBBB...
370 @ B.BBBB.....BB...........
380 @ ..B..B.....BBBB.........
390 REM
400 REM
410 DESIGN 0,14*64 : REM SPRITEDEFINITIE IN BLOK 14
420 @ .............BBB....B.B.
430 @ ............BB.BB....B..
```

```
440 @ ..BBB.B....BBBBBBBB.....
450 @ .BBBBB.....BBBBBBB..BB..
460 @ ..BBB.B.....BBBBBBBBBB..
470 @ .............BBB........
480 @ ............BBB.........
490 @ ............BBB.........
500 @ ............BBB.........
510 @ ............BBBB........
520 @ ............BBBBB.......
530 @ ............BBB.BB......
540 @ ...........BBBB.........
550 @ ..........B.BBBB........
560 @ .........B..BB.BB.......
570 @ ........B...BB.BB.......
580 @ .......B..BBB..BB.......
590 @ ......BB.BB....BB.......
600 @ .BBBBBBB.BBB...BB.......
610 @ B.BBBB.....BB..BB.......
620 @ ..B..B.........BBBB.....
630 PRINT CHR$ (30) CHR$(147) : REM GROEN + BEELDSCHERM WISSEN
640 PRINT AT (0,9) DUP(CHR$(18) + " ",40):REM GRAS TEKENEN
650 REM
660 REM
670 REM **************
680 REM **          **
690 REM **  LOPEN   **
700 REM **          **
710 REM **************
720 REM
730 MMOB 0,0,100,0,100,1,0 : REM SPRITE POSITIONEREN
740 S = 10 : REM SNELHEID
750 K = 1 : REM STARTKLEUR
760 FOR X=1 TO 300 STEP G
770 K = K+.3: REM KLEUR VERANDEREN
780 IF K=16 THEN K=1
790 REM ---------------
800 MOB SET 0,13,K,0,0
810 RLOCMOB 0,X,101,1,0 : REM EERSTE SPRITE BEWEGEN
820 FOR Y=1 TO 20 : NEXT Y
830 MOB OFF 0 : REM EERSTE SPRITE UITSCHAKELEN
840 REM ---------------
850 MOB SET 0,14,K,0,0
860 RLOCMOB 0,X+S/2,101,1,0 : REM TWEEDE SPRITE BEWEGEN
870 FOR Y=1 TO 20 : NEXT Y
880 MOB OFF 0 : REM TWEEDE SPRITE UITSCHAKELEN
890 REM ---------------
900 NEXT X
910 COLOUR 9,9
```

Dit voorbeeld laat u het werken met meer spritedefinities tegelijkertijd zien. De regels 830 en 880 kunnen uiteraard weer weggelaten worden. Ze zijn er slechts voor een beter begrip. In regel 740 kunt u de snelheid van het mannetje veranderen. Probeert u de mogelijkheden. Laat u eens een zwerm bijen langs vliegen of plaatst u bomen langs de kant van de weg. Misschien bestuurt u het mannetje nog wel eens met het toetsenbord of met de joy-stick, wie weet?

14.2.4 DETECT

Format	DETECT k
Parameter	k : soort van de voorbereiding op het samentreffen (k=0: sprite-sprite / k=1: sprite-achtergrondteken)
Voorbeeld	DETECT 1
Functie	Voorbereiding van het testen op een samentreffen

Deze instructie bereidt het detecteren van een samentreffen van sprites met iets anders voor. Dat klinkt gecompliceerd en dat is het ook. Zoals u uit de inleiding weet, zijn er in de registerzetting van de VIC twee geheugenplaatsen, die samentreffingen registreren. Een van de twee is voor de samentreffingen van sprites onder elkaar, de andere voor die tussen een sprite en een achtergrondteken (in de tekstmodus bijvoorbeeld met letters). Deze registers worden gezet, als er een samentreffen plaatsgevonden heeft. Maar de inhoud blijft behouden tot ze uitgewist wordt. Dat wil zeggen: als u dit register na een vraag of er een samentreffen plaatsgevonden heeft, niet uitwist, lijkt het alsof deze oude gebeurtenis nog actueel zou zijn, hoewel er op dat moment geen samentreffen plaatsvindt.

Om dit te vermijden is de DETECT-instructie noodzakelijk. Deze wist het register van samentreffingen uit en bereidt daarmee een nieuwe vraag voor. Deze instructie zou daarom na elke vraag op samentreffingen moeten staan, zodat de volgende vraag voorbereid wordt. Als er een samentreffen plaatsgevonden heeft, is het niet genoeg om deze instructie nog een keer op te geven, u moet in de loop van het programma (ook als tussendoor het programma al ten einde was -zie voorbeeld) tot de volgende vraag deze instructie op z'n minst twee keer opgegeven hebben. Anders registreert u foutief een tweede samentreffen. Dit berust op bepaalde hardware-eigenschappen

en valt buiten het kader van dit boek. Het gebruik is niet zo eenvoudig als u het zich zou wensen, maar met wat oefening krijgt u het wel voor elkaar. De enige parameter in deze instructie is ervoor verantwoordelijk, dat er vastgelegd wordt of een detecteren voor een samentreffen van een sprite-sprite of een sprite-achtergrondteken moet worden voorbereid. Meer hierover leest u in de volgende instructie-uiteenzetting.

14.2.5 CHECK

Format	CHECK (n1,n2) of CHECK (n)
Parameter	n1: nummer van de eerste sprite, die op een samentreffen met een andere sprite getest moet worden (0-7)
	n2: nummer van de tweede sprite, die op een samentreffen met een andere sprite getoetst moet worden (0-7)
	n : nummer van de sprite, die op een samentreffen met een achtergrondteken getoetst moet worden (0-7)
Voorbeeld	CHECK (0,7) of CHECK (1)
Functie	Het detecteren van een samentreffen

Zo zijn we bij de laatste sprite-instructie aangekomen die Simon's BASIC ons geeft: CHECK. De instructie CHECK heeft weer het karakter van een functie, net als de instructies SIN, SQR, LIN etc. Hij wordt net zoals de al bekende instructie TEST gebruikt, die toetst of er op een plaats in een grafiek een punt geplaatst is. CHECK moet vaststellen of er een samentreffen van een sprite met een ander object (sprite of achtergrondteken) plaatsgehad heeft. Hij wordt met de instructie DETECT ingeleid. Als er op het tijdstip van de verwezenlijking geen samentreffen geregistreerd is, krijgt CHECK de waarde 1. Deze waarde kunt u natuurlijk willekeurig aan een variabele toekennen, of er berekeningen mee uitvoeren. Bijvoorbeeld:

```
A = 2 * CHECK(0) + 1
```

Als er een samentreffen geregistreerd werd, wordt aan de functie CHECK de waarde 0 toegekend. In de voorafgaande DETECT had u al aangegeven of het om een samentreffen sprite-sprite of

sprite-achtergrondteken gaat. Deze informatie wordt bij het gebruik van de CHECK-instructie weer opgepakt. Werd dus het eerste geval voorbereid, dan moet de syntax van de daaropvolgende CHECK luiden:

CHECK(n1,n2)

Met deze parameters geeft u aan tussen welke twee sprites u een samentreffen wilt controleren. Andere sprites, die elkaar eventueel ook getroffen hebben, worden daarbij over het hoofd gezien. Alleen wanneer deze twee objecten, waarvan u de nummers met n1 en n2 gedetermineerd hebt, elkaar aangeraakt hebben, wordt CHECK 0. Als u enkel en alleen wilt toetsen of een sprite in aanraking is geweest met een willekeurige andere sprite, kiest u n1=n2. In dat geval wordt elke aanraking van deze sprite met een andere sprite geregistreerd. Hetzelfde geldt, als u het tweede geval met DETECT 1 hebt voorbereid. Hier luidt de syntax dan:

CHECK(n)

waarbij n het nummer van de sprite voorstelt, waarvan het samentreffen met de achtergrond getoetst moet worden. De informatie die het handboek daarover geeft, is onjuist. Belangrijk in dit verband is het feit, dat (zoals bij de instructie AT) het eerste haakje achter het woord CHECK eigenlijk nog bij de instructie hoort en niet door een spatie mag worden afgezonderd. Dus:

CHECK(2,2) en niet: CHECK (2,2)

```
100 REM ***********************
110 REM **                   **
120 REM **  CHECK- -VOORBEELD **
130 REM **                   **
140 REM ***********************
150 REM
160 COLOUR 0,0 : REM VENSTER- ACHTERGRONDKLEUR = ZWART
170 DESIGN 1,13*64 : REM SPRITEDEFINITIE IN BLOK 13
180 @ ............
190 @ ............
200 @ ............
210 @ ............
220 @ ............
230 @ .C..........
240 @ CCC.........
250 @ CCCBB.......
260 @ CCBDBBDDDB..
270 @ CCCBBDBBBDBB
280 @ CCBD........
290 @ CCC.........
300 @ .C..........
310 @ ............
```

```
320 @ ............
330 @ ............
340 @ ............
350 @ ............
360 @ ............
370 @ ............
380 @ ............
390 REM
400 REM
410 DESIGN 0,14*64 : REM SPRITEDEFINITIE IN BLOK 14
420 @ ........................
430 @ ........................
440 @ ........................
450 @ ........................
460 @ ........................
470 @ .......BBB..............
480 @ ...........B............
490 @ ............B...........
500 @ .............B..........
510 @ ............BBBBB.......
520 @ ......BB...BBBBBB.......
530 @ .....BBB...BBB..BB......
540 @ ....BBBBB..BBB..BBB.....
550 @ ..BBBBBBBBBBBB..BBBB....
560 @ ..BBBBBBBBBBBBBBBBBB....
570 @ ..BBBBBBBBBBBBBBBBBB....
580 @ ...BB..BBBBBBBB..BB.....
590 @ BBBB.BB.BBBBBB.BB.BB....
600 @ .....BB........BB.......
610 @ ........................
620 @ ........................
630 REM
640 MMOB  0,0,100,0,100,0,0 : REM EERSTE SPRITE POSITIONEREN
650 MMOB  1,300,100,300,100,1,0 : REM TWEEDE SPRITE POSITIONEREN
660 DETECT 0 : REM BOTSREGISTER WISSEN
670 REM
680 REM **************
690 REM **          **
700 REM **  RIJDEN  **
710 REM **          **
720 REM **************
730 REM
740 FOR X=1 TO 300 STEP 4
750 CMOB 2,7 : REM KLEUREN VOOR MULTICOLOR-SPRITE
760 REM ---------------
770 MOB SET 0,14,9,0,0
780 RLOCMOB 0,X,100,0,0 : REM EERSTE SPRITE BEWEGEN
790 REM ---------------
800 CMOB 7,2 : REM KLEUREN VOOR MULTICOLOR-SPRITE
```

```
810 MOB SET 1,13,11,0,1
820 RLOCMOB 1,320-X,40*SIN(X/30+1.3)+100,1,0 : REM TWEEDE SPRITE
    BEWEGEN
830 REM ---------------
840 DETECT 0 : REM AFTASTEN VAN SPRITE-SPRITE-BOTSINGEN VOORBEREIDEN
850 IF CHECK(0,1)=0 THEN GOTO 870
860 NEXT X
870 FOR Z=1 TO 15
880 COLOUR Z,Z-1
890 FOR S=1 TO 30 : NEXT S : REM WACHTLUS
900 NEXT Z
910 HIRES 13,14
```

14.2.6 Testopdrachten

1. Waar moet u op letten bij het gebruik van DETECT en CHECK?

   A) DETECT moet na een samentreffen, dat plaatsgehad heeft, een keer ingevoerd worden om het register van de samentreffingen uit te wissen.
   B) Door de parameter van de DETECT-instructie wordt de syntax van CHECK bepaald.
   C) CHECK(1,1) is ongeoorloofd.
   D) CHECK(9) is ongeoorloofd.
   E) Als een samentreffen heeft plaatsgehad, krijgt CHECK de waarde 1.

2. Welke syntax is juist?

   A) MMOB 1,30,40,50,60,2,70
   B) MOB OFF
   C) RLOCMOB 3,70,3
   D) CHECK (2,3)

15 MUZIEK

Binnen een paar minuten duiken we onder in het geheimzinnige rijk der klanken. Voor u staat een volledige, goed toegeruste en driestemmige musicsynthesizer. Denkt u maar niet, dat de muziek van de radio een andere oorsprong heeft. Met een beetje bekwaamheid zult u de mooiste melodieën en het meest krakende geluid aan uw computer kunen ontlokken. Uiteraard is Simon's BASIC geen vervanging voor een echt synthesizer-programma, dat alle mogelijkheden uitputtend behandelt, zoals SYNTHIMAT.

15.1 Hardwarevoorzieningen

Om de afzonderlijke instructies voor de klankproduktie te begrijpen, moet het u eerst duidelijk zijn op welke manier een klank is opgebouwd. Er zijn zeer complexe geluiden en klankkarakteristieken. Om een eigen klank te produceren, moeten we weten uit welke componenten zo'n klank samengesteld is. Allereerst is er de toonhoogte. Wat is er nu eigenlijk de oorzaak van, dat een klank een bepaalde toonhoogte heeft? Zoals u uit de natuurkunde wel weet, is geluid een zeer snelle beweging van de lucht, die we met onze oren waarnemen. De eenvoudigste golf die we ons kunnen voorstellen, is een sinusvormige, volledig gelijkmatige golf.

Het is mogelijk om zo'n geluidsgolf te produceren, waardoor de lucht sinusvormig golft. Veel mensen hebben dit al een keer uit verschillende apparaten gehoord. De verschillen in toonhoogte komen tot stand, doordat de afzonderlijke toppen en dalen van zo'n golf meer of minder dicht op elkaar volgen. Dat wil zeggen: ze komen in een sneller of langzamer ritme op ons af. Het aantal op ons afkomende golven per seconde noemen we de frequentie. We nemen dit waar als de toonhoogte. Naast de frequentie is ook de geluidssterkte een karakteristiek gedeelte van een klank. Die komt door het hoogteverschil van toppen en dalen van een golf (amplitude) tot stand.

De derde factor die de karakteristiek van een klank bepaalt, is de vorm van een golf. Er zijn zeer uiteenlopende mogelijkheden. Uw Commodore 64 geeft u vier mogelijkheden, om de golf van een klank te bepalen.

1. driehoek

2. zaagtand

3. rechthoek

4. ruis

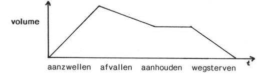
Daarbij kunnen deze golfvormen met elkaar gemengd en logisch bewerkt worden. Een essentiële eigenschap van de klanken hangt samen met het spectrum van hun boventonen. Geen klank behalve die van de zuivere sinusvorm volgt exact het getekende verloop. De meeste golfbewegingen trillen alsof ze met een bibberige hand getekend zijn. Deze subfrequenties horen we eveneens en zij zijn karakteristiek voor een klank. Slechts aan de hand van deze boventonen kunnen we een viool van een fluit onderscheiden. De golfvormen die uw computer u biedt, hebben verschillende boventoonspectra en daarom kunnen we ze onderscheiden. Nu is het mogelijk om gedeelten van deze hogere subfrequenties uit of af te snijden. Dat wordt met zogenaamde filters gedaan. Zulke filters zitten ook in elke moderne versterker: de bas/treble-knop. Met deze knop zijn tonen weg te filteren waardoor het geluid anders klinkt. Uw

computer is uitgebreid van zulke mogelijkheden voorzien. Ze bepalen de frequentie (toonhoogte) van een klank, de golfvorm, geluidssterkte en de frequenties, die eruit gefilterd moeten worden.

Maar een aspect ontbreekt nog dat het totale beeld afrondt: het verloop van een klank. Geen enkele klank is er plotseling en dan direkt weer weg. Er zijn klanken die langzaam sterker worden en weer afnemen, of schoten die als een zweep klinken en een beetje nagalmen, enzovoort. Uw computer maakt het u mogelijk om dit verloop van een klank te programmeren. Het verloop wordt daarbij in vier fasen ingedeeld:

1.) aanzwel-fase	ATTACK
2.) afval-fase	DECAY
3.) stabiele fase	SUSTAIN
4.) wegsterf-fase	RELEASE

Grafisch is dat als volgt voor te stellen:



Het verloop werkt als volgt: allereerst weerklinkt de klank. U voert de tijd in, die de klank nodig heeft om tot de hoogste (eveneens aangegeven) geluidssterkte aan te zwellen. Op deze hoogste geluidssterkte gekomen, valt het volume weer af tot het sustainniveau dat een door u bepaalde tijd gehandhaafd blijft en daarna begint de wegsterf-fase, waarvan u de duur ook aangegeven hebt. De totale verloopdefinitie ("de omhullende golfvorm") wordt natuurlijk bij het begin van een muziekstuk aangegeven en blijft tot haar verandering van kracht. Uw computer heeft drie onafhankelijke stemmen, die ook tegelijkertijd kunnen klinken. Voor alle drie de stemmen kunt u afzonderlijk de frequentie, de golfvorm en het verloop aangeven. De geluidssterkte en het filter gelden echter gemeenschappelijk voor alle drie de stemmen. Er zijn nog andere functies, die hier echter niet behandeld kunnen worden, omdat Simon's BASIC ze niet gebruikt. Voor verdere informatie kunt u in uw gebruikershandboek kijken, of in het boek dat specifiek de geluidsmogelijkheden van de CBM 64 behandelt:

HET MUZIEKBOEK VOOR DE COMMODORE 64 van DATA BECKER/NEDERLANDS*.

15.2 VOL

Format	VOL 1
Parameter	- I : geluidssterkte (0-15)
Voorbeeld	VOL 15
Functie	Instellen van de geluidssterkte van de te spelen klanken

De eerste van de in totaal vijf klankinstructies is VOL (van VOLume). Daarmee hebben we de mogelijkheid om de geluidssterkte van alle drie de klankgeneratoren tegelijkertijd vast te leggen of te veranderen. Omwille van de doelmatigheid gebeurt dat aan het begin van een muziekprogramma. U kunt deze geluidssterkte natuurlijk ook tijdens het programma of zelfs tijdens een toon veranderen, wat speciale effecten tot gevolg heeft. Helaas is het door de hardware niet mogelijk elke stem een eigen geluidssterkte te geven. Als u I=0 laat worden, zwijgen alle drie de stemmen, terwijl I=15 de grootste geluidssterkte aangeeft.

Voorbeeld:

```
100 REM ********************
110 REM **                **
120 REM **  VOL-VOORBEELD **
130 REM **                **
140 REM ********************
150 REM
160 REM
170 REM       **********
180 REM       **      **
190 REM       ** TOON **
200 REM       **      **
210 REM       **********
220 REM
230 VOL 15
240 ENVELOPE 1,1,8,10,10
250 WAVE 1,00100000
260 MUSIC 150,"S1c2I"
270 PLAY 2
280 REM
290 REM    *****************
300 REM    **             **
310 REM    ** AANZWELLEN  **
```

```
320 REM  **               **
330 REM  *****************
340 REM
350 FOR X=0 TO 15
360 VOL X : REM VOLUME
370 FOR W=1 TO 80 : NEXT W
380 NEXT X : REM VOLUME LATEN AANZWELLEN
390 PAUSE 2
400 VOL 0 : REM VOLUME OP 0 ZETTEN
410 PAUSE 2
420 REM
430 REM      ***********
440 REM      **       **
450 REM      **  ECHO **
460 REM      **       **
470 REM      ***********
480 REM
490 REM
500 FOR X=15 TO 0 STEP -1
510 FOR Y=0 TO X
520 VOL Y : REM VOLUME
530 FOR W=X TO 35 : NEXT W : REM WACHTLUS
540 NEXT Y : REM VOLUME LATEN AANZWELLEN
550 NEXT X : REM VOLUME LATEN AFNEMEN
```

De regels 240 tot 270 zullen u zeker nog onbekend voorkomen. Ze zijn voor het produceren van een lange klank. We zullen de afzonderlijke instructies stap voor stap doornemen en uitleggen. In regel 260 worden een paar vreemde regels tussen aanhalingstekens aangegeven. Dit is voor u een ondersteuning bij het overnemen van het programma. De hoofdletters betekenen bij de instructie MUSIC van nu af aan controletekens. Deze tekens zullen ook bij u op het beeldscherm als inverse hoofdletters getoond worden, voor het geval dat u de andere karakterset hebt ingeschakeld (dus na het gelijktijdig indrukken van de toetsen SHIFT en C=). Daarmee worden steeds de volgende controletekens bedoeld:

S:	SHIFT CLR/HOME
I:	F2
J:	F4
K:	F6
L:	F8
E:	F1
F:	F3
G:	F5
H:	F7

Deze tabel geldt voor alle geluidsvoorbeelden in dit boek. Als u dus niet meer weet wat er bedoeld wordt, kunt u het hier opzoeken. Alle kleine letters in deze MUSIC-instructies zijn normale toetsen. U zou regel 260 in dit programma dus ook zo kunnen schrijven:

260 MUSIC 150, CHR$(147) + "1c2" + CHR$(137)

15.3 WAVE

Format	WAVE st, rrsdtrsg
Parameter	st : stem ,waar de golfvorm voor wordt aangegeven (1-3) rrsdtrsg : rij van acht keer 1 of 0 voor het aan- (1) en uitschakelen (0) van bepaalde functies
Voorbeeld	WAVE 2,10000101
Functie	Instellen van de golfvorm van de te spelen klanken

We zitten al midden in de klanksynthese. WAVE biedt u de mogelijkheid om de afzonderlijke, door de computer beschikbaar gestelde golfvormen, telkens voor een van de drie mogelijke stemmen vast te leggen. Naast deze taak kunt u nog een serie andere functies bepalen. Maar eerst iets over de op te geven parameters: de eerste van de twee aan te geven waarden bepaalt de stem, waarvoor uw golfdefinitie moet gelden. U kunt de karakteristiek telkens voor alle drie de stemmen verschillend bepalen. De tweede parameter zou u op het eerste gezicht een beetje verwarrend kunnen voorkomen. U kunt met de instructie WAVE acht verschillende functies uitkiezen, die later beschreven worden. Al deze functies kennen slechts twee mogelijkheden: aan of uit. U hoeft verder niets anders te doen dan na elkaar telkens een 1 voor een ingeschakelde functie (bijvoorbeeld golfvorm driehoek) of een 0 voor een uitgeschakelde functie in te toetsen. Voor de acht mogelijke functies geeft u acht enen of nullen achter elkaar. Met deze instructie programmeert u direkt een register van de SID (soundprocessor). Het zijn de registers 4, 11 en 18 voor de drie stemmen (zie gebruikershandboek, pp. 155-157). De acht afzonderlijke waarden die u moet invoeren, stellen de acht bits van dit register voor, die de verschillende taken op zich nemen. Ze worden verklaard na de functies van de bits.

bit.....functie
7......ruis
6......rechthoek
5......zaagtand
4......driehoek
3......testbit
2......modulatie
1......synchronisatie
0......gatebit

- **bit 0 - gatebit:** Deze bit is er ten behoeve van de initialisering en de beëindiging van een klank. Als hij op 1 geplaatst wordt, begint de met de andere registers ingestelde klank volgens zijn van te voren bepaalde verloop te klinken. De toon wordt in de sustain-fase gehouden totdat deze bit weer naar 0 teruggeplaatst wordt. Als dat gebeurd is, ebt hij weg, zoals dat door release gedefinieerd is. Normaal wordt deze bit automatisch door Simon's BASIC bestuurd. Hij moet daarvoor in een WAVE-aanwijzing geplaatst worden, als er tussendoor, terwijl er een toon klinkt, een parameter door WAVE moet worden veranderd, omdat de toon anders uitgeschakeld wordt. Omgekeerd kunt u deze bit voor de uitschakeling van een klank gebruiken.

- **Bit 1 - synchronisatie:** Als u tegelijkertijd meer klanken op meer stemmen, dus op meer oscillatoren laat spelen, zult u zien dat er, afhankelijk van de klank, verschillende trillingen te horen zijn. Deze trillingen komen tot stand, doordat er bijvoorbeeld op een bepaald tijdstip een trilling-dal van de ene stem met een trilling-top van de andere samenvalt. In dit geval heffen de beide golven elkaar natuurlijk op en hoort u niets. Kort daarop echter zal een keer een top op een andere trilling-top stuiten, wat tot een wederzijdse versterking leidt. De klank wordt sterker. Deze beide extrema wisselen voortdurend. Daarom hoort u de trilling van een klank. Soms echter is dit effect ook storend. Dan plaatst u gewoon de synchronisatie-bit en dan trillen de beide klanken in dezelfde fase. Bit 1 heeft bij de drie stemmen verschillende taken:

stem	synchronisatie
1.............	stem 1 met 3
2.............	stem 2 met 1
3.............	stem 3 met 2

- **Bit 2 - ringmodulatie:** Als bit 2 een 1 is, wordt de ringmodulatie ingeschakeld. Deze functie veroorzaakt een met synchronisatie vergelijkbaar effect. Alleen wordt hierdoor ook het volume beïnvloed. Het resultaat is een soort wow-wow-effect door het

variëren van de frequentie van de ene stem tegen de statische frequentie van een andere. Er kan een groot scala van niet-harmonische toonstructuren door worden bereikt. De toepassing is geschikt voor bel- en gonggeluiden en diverse speciale effecten. Het beste werkt de keuze van een driehoeksgolf als variabele frequentie. Die wordt dan vervangen door de gemoduleerde golfvorm die uit de combinatie van de oorspronkelijke driehoeksgolf en de andere gedefinieerde golfvorm ontstaat. Net als bij synchronisatie bepaalt de voor de uitvoer gekozen stem de keuze van de als statische frquentie gebruikte stem. Dezelfde tabel is hier dus van kracht.

- **Bit 3 - testbit:** Als samen met de ruisgenerator nog een andere trillingsvorm van dezelfde stem uitgekozen zou worden, kan het gebeuren dat de ruisgenerator blokkeert. De blokkade kan door deze bit weer opgeheven worden.

- **Bit 4 - driehoekstrilling:** Als deze bit geplaatst wordt, heeft u de driehoekstrilling uitgekozen. Deze lijkt op de zuivere sinustrilling, die geen harmonische boventonen heeft. Het lijkt op de toon van een fluit.

- **Bit 5 - zaagtandtrilling:** De zaagtandtrilling heeft veel oneven en even boventonen en klinkt zeer hard (zoals een viool).

- **Bit 6 - rechthoektrilling:** Bij de rechthoektrilling kunt u extra de verhouding van de duur van de dalen en toppen variëren, de zogenaamde pulsverhouding. Simon's BASIC geeft geen mogelijkheid dit aan de hand van een instructie te laten zien, hoewel de pulsverhouding voor het gebruik van de rechthoektrilling een keer moet worden bepaald. Daarom moeten we ons met twee POKEs behelpen: ST is een beschermd woord!!

  POKE54272-5+7*ST,LB : POKE54272-4+7*ST,HB kan niet

  Daarbij draagt de variabele ST het nummer van de stem waarvan de pulsverhouding veranderd moet worden. HB stelt de high-byte en LB de low-byte van de in te voeren waarde voor.

- **Bit 7 - ruis:** Met het plaatsen van deze bit wordt de ruisgenerator van de betreffende stem ingeschakeld. Deze modus is zeer veelzijdig. U kunt er het ruisen van de zee, schoten, gesis en nog veel meer mee produceren.

Een opmerking bij de bits 4-7: het is in de praktijk mogelijk meer trillingsvormen tegelijkertijd uit te kiezen. Maar u moet er dan op letten, dat het resultaat niet de som van alle vormen is, maar een logische AND-verbinding van de componenten (zie voorbeeld).

Voorbeeld:

```
100 REM ********************
110 REM **                **
120 REM ** WAVE-VOORBEELD **
130 REM **                **
140 REM ********************
150 REM
160 REM
170 REM      **********
180 REM      **      **
190 REM      ** TOON **
200 REM      **      **
210 REM      **********
220 REM
230 VOLUME 15
240 ENVELOPE 1,10,10,10,10
250 MUSIC 250,"S1c2I"
260 PLAY 2
270 REM
280 REM  ****************
290 REM  **            **
300 REM  **  WAVEVORM  **
310 REM  **            **
320 REM  ****************
330 REM
340 REM ZAAGTAND:
350 WAVE 1,00100001
360 PAUSE 2
370 REM RUIS:
380 WAVE 1,10000001
390 PAUSE 2
400 REM DRIEHOEK/ZAAGTAND:
410 WAVE 1,00110001
420 PAUSE 2
430 REM DRIEHOEK:
440 WAVE 1,00010001
450 PAUSE 2
460 REM DRIEHOEK/BLOKGOLF:
470 WAVE 1,01010001
480 PAUSE 2
490 REM BLOKGOLF/ZAAGTAND:
500 WAVE 1,01100001
510 PAUSE 2
520 REM DRIEHOEK/BLOKGOLF/ZAAGTAND:
530 WAVE 1,01110001
540 PAUSE 2
```

15.4 ENVELOPE

Format	ENVELOPE st, a, d, s, r
Parameter	st: stem, waarvoor de omhullende golfvorm wordt aangegeven (1-3)
	a : attack-duur van de aanzwel-fase (0-15)
	d : decay-duur van de zachter-worden-fase (0-15)
	s : sustain-geluidssterkte in de aanhoud-fase (0-15)
	r : release-duur van de uitsterf-fase (0-15)
Voorbeeld	ENVELOPE 3,9,6,10,15
Functie	Instellen van de omhullende golfvorm van de te spelen klanken

Nu komen we bij het echt creatieve gedeelte van de klankproduktie, de bepaling van het verloop, de omhullende golfvorm. We hadden al in de inleiding van dit hoofdstuk beschreven, wat onder een omhullende golfvorm verstaan wordt: het vastleggen van het verloop van een klank ten opzichte van de geluidssterkte. Deze eigenschap kan voor alle drie de stemmen apart gedefinieerd worden. Met behulp van de vier parameters, die telkens voor de afzonderlijke klankfasen verantwoordelijk zijn, hebt u een dermate groot spectrum van mogelijkheden, dat het u moeilijk zal vallen een echt overzicht te krijgen. Bij de overweging hoe een klank voorgesteld moet worden, zijn er twee mogelijkheden om tot een aantrekkelijk geluid te komen.

1. U probeert net zo lang de afzonderlijke parameters te veranderen, totdat u een mooie klank tot stand heeft gebracht. Dat moet u trouwens steeds doen om te kunnen zien welk effect de parameters hebben en welke mogelijkheden er zijn. Deze methode is vooral bij het produceren van effecten te gebruiken.

2. Wanneer u een bepaalde klank (bijvoorbeeld van een fluit) wilt nabootsen, moet u eerst analyseren hoe het klankverloop van de natuurlijke klank is om daarna de daarmee overeenkomende parameters te plaatsen. Dit is een veel doelbewustere weg. Een fluitklank heeft een lange aanzwel-fase (attack wordt groot) en wordt op geluidssterkte gehouden zonder merkbaar verval

beneden de maximale geluidssterkte (sustain wordt groot). Wanneer het verval inzet, gebeurt dat heel langzaam (decay groot). De wegsterf-fase is nauwelijks hoorbaar, omdat de klank meestal zeer abrupt eindigt (release klein).

Als voorbeeld wordt hier een klein programma gegeven, waarin u in regel 290 de afzonderlijke parameters van de ENVELOPE-instructie moet veranderen om zo een overzicht van de verschillende mogelijkheden van de omhullende golfvorm te krijgen. Verander ook de golfvorm in regel 410. U kunt een keer de ruis kiezen, waarbij de omhullende golfvorm heel goed te horen is.

```
100 REM ************************
110 REM **                    **
120 REM **  ENVELOPE-VOORBEELD **
130 REM **                    **
140 REM ************************
150 REM
160 REM
170 REM      ****************
180 REM      **            **
190 REM      **  OMHULLENDE **
195 REM      **  GOLFVORM   **
200 REM      **            **
210 REM      ****************
220 REM
230 REM VOORBEELD:
240 REM VERANDERT U DE PARAMETERS EENS!
250 REM
260 REM      ST! A! D! S! R
270 REM ------+--+--+--+---
280 REM       !  !  !  !
290 ENVELOPE 1,10,10, 5, 9
300 REM ------+--+--+--+---
310 REM
320 REM
330 REM          **********
340 REM          **      **
350 REM          ** TOON **
360 REM          **      **
370 REM          **********
380 REM
390 VOLUME 15
400 REM ZAAGTAND:
410 WAVE 1,00100001
420 MUSIC 50,"S1c2I"
```

```
430 PLAY 2
440 PAUSE 1 : REM UITSCHAKELEN
450 WAVE 1,00100000
```

Voor de instructies in de regels 420 en 430 en vooral de schrijfwijze van de string in regel 420 kunt u de opmerkingen bij het VOL-voorbeeld in paragraaf 15.2 nalezen.

15.5 MUSIC

Format	MUSIC td, str
Parameter	td : klankduur of tijdmaat, waarmee het stuk moet worden gespeeld (0-255)
	str: stringvariabele of "reeks tekens", die het klankverloop vastlegt
Voorbeeld	MUSIC 8,A$ of MUSIC 10,"...tonen..."
Functie	Vastleggen van de melodie van een muziekstuk

We komen bij een instructie, die het mogelijk maakt om een bepaalde klankvolgorde, die u graag wilt spelen, vast te leggen. De computer onthoudt deze klankvolgorde en speelt die af na een PLAY-instructie. Met td geeft u daarbij aan met welke snelheid hij de noten achter elkaar moet spelen, dus hoe snel een stuk in z'n geheel moet zijn. In de muziek zijn er daarvoor verschillende tempi-aanwijzingen, bijvoorbeeld allegro, andante of presto. Daarbij is de basis-eenheid 1/60 seconde. Bij td=0 heeft een hele noot (1/1) de duur van 2/60 seconde. Een halve noot duurt dan precies 1/60 seconde. Nog kleinere notenwaarden duren daarbij niet korter. Bij td=1 ligt de duur van 1/1 noot al bij 5/60=1/12 seconde. Bij de poging om 1/4 noot te spelen, zult u enkele oneffenheidjes te horen krijgen, omdat 5 niet zonder rest door 4 deelbaar is. In het algemeen laat zich de tijdsduur van een hele noot ongeveer door de volgende formule bepalen (in seconden):

$$z = td/12 \qquad \text{of} \qquad z = 5*td/60$$

Deze formule levert telkens slechts waarden bij benadering, die van keer tot keer kunnen verschillen. td=0 valt volkomen buiten de boot. De afzonderlijke andere notenwaarden zijn fracties of veelvouden van

de uitkomst. Om een zo precies mogelijke fractie, dus een zo precies mogelijk gespeelde kwart, achtste etc. te krijgen, moet u waarden voor td plaatsen, die door 4 of 8 zonder rest deelbaar zijn.

Nu de eigenlijke definitie van de noten, die in de volgende string kan worden aangegeven. Het beste kunt u voor dit doel steeds uw apparaat naar de kleine letters/hoofdletters schakelen, met de SHIFT- en de C=-toets tegelijkertijd ingedrukt. Mocht dit niet functioneren (hetgeen bij Simon's BASIC wel eens voorkomt), dan voert u in:

PRINT CHR$(9)

Hierdoor zijn de in te voeren tekens beter te onderscheiden en komen ze met de tekens overeen, die in de voorbeeldprogramma's aangegeven zijn. Aan het begin van de string moet u bepalen voor welke stemmen u de volgende noten van plan bent te programmeren. Dat voert u in, doordat u (na invoer van de aanhalingstekens) op SHIFT + CLR/HOME drukt. Er verschijnt een inverse hartje of een inverse S. Direkt daarna volgt het nummer van de op te roepen stem (1-3). Deze uitdrukking kunt u willekeurig vaak in uw muziekstuk gebruiken om op een andere stem over te schakelen. Dit kan bijvoorbeeld nuttig zijn, als u in de drie stemmen andere golfvormen of envelopes hebt uitgekozen en midden in het stuk de klank wilt wisselen. Dan begint de eigenlijke klankprogrammering: als volgende parameter geeft u de te spelen klank op. Dit verdeelt u in twee delen. Allereerst de naam van de klank, daarna het octaaf. Voor de naam van de klank worden de gebruikelijke letters van de toonladder op de Amerikaanse schrijfwijze ingevoerd:

C D E F G A B

Als u een toon met een halve klank wilt verhogen (dus van een cijferkruis voorzien), wat overeenkomt met de verlaging van de daarboven liggende toon, drukt u de betreffende letter samen met SHIFT in. Daarbij geeft u een cijfer van 0-7 voor het octaaf waarin de toon moet vallen aan. Ter oriëntatie: de A met een frekwentie van 440 Herz ligt in het vierde octaaf. U moet dus invoeren: A4. De hoogste toon die u kunt invoeren, is de verhoogde A van het 7e octaaf: a7. Om de definitie van een noot af te sluiten, geeft u de duur van de noot aan. Deze wordt door middel van de functietoetsen ingevoerd. Daarbij krijgen de afzonderlijke functietoetsen de volgende waarden voor de noten:

toets....nootwaarde.....beeldschermteken
f1........1/16.........inverse-hoofdletter E
f3........1/8 ....inverse-hoofdletter F
f5........1/4 ....inverse-hoofdletter G
f7........1/2 ....inverse-hoofdletter H
f2........1/1 ....inverse-hoofdletter I
f4........2/1 ....inverse-hoofdletter J

f6........4/1 ....inverse-hoofdletter K
f8........8/1 ....inverse-hoofdletter L

Met beeldschermteken is het teken bedoeld, dat op het beeldscherm verschijnt, als u (nadat u een aanhalingsteken ingevoerd hebt) op de betreffende toets drukt. Daarmee heeft u een toon gedefinieerd. De volgende, die weer uit drie tekens bestaat, wordt er direkt achter gevoegd. Als u bijvoorbeeld een 3/2 noot (1 1/2) wilt spelen, kunt u ook meer functietoetsen achter elkaar als lengte-informatie invoeren, die dan steeds opgeteld worden. In dit geval van een 3/2 zou dat zijn: f7 f7 f7 of f2 f7. Hier wordt natuurlijk de toetsenvolgorde aangegeven, niet de beeldschermtekens. Gewoonlijk worden de afzonderlijke noten samen (legato) gespeeld. Als u een kleine rust wilt hebben, voert u in: SHIFT CLR/HOME met een daarop volgende g. Dit moet u eveneens doen als u de stem wilt verwisselen en de voorafgaande stem niet wilt laten verderklinken. Deze invoer schakelt de klank uit. Als u achter deze instructie een (of meer) nootwaarde aangeeft, onstaat er een pauze met de aangegeven duur. Zo kunt u zeer eenvoudig een liedje programmeren. Grotere composities definieert u door meer strings samen te voegen. Zo kunt u ook herhalingen invoeren. Het voorbeeld vindt u bij PLAY.

15.6 PLAY

Format	PLAY m
Parameter	m: speelmodus: m=1: op het einde van het spel wachten m=2: spelen en rekenen
Voorbeeld	PLAY 1
Functie	Spelen van de met MUSIC vastgelegde klankvolgorde

Eindelijk kunnen we iets van onszelf produceren. Met de PLAY-instructie is het mogelijk de volgorde van de klanken, die we in de MUSIC-instructie hebben vastgelegd, met de eigenschappen, die de overige instructies determineren, te laten spelen. DE PLAY-instructie kent twee modificaties, die met de enige parameter m kunnen worden gekozen. Als u m=1 plaatst, wacht het programma tot de muziek afgelopen is. Dat wil zeggen: alle instructies, die in uw programma achter PLAY 1 komen, worden pas na beëindiging van de klank uitgevoerd. Dan is het programma ook slechts door RUN/STOP RESTORE af te breken. Als u in plaats daarvan PLAY 2 invoert, worden direkt na het begin van de muziek de erop volgende instructies

uitgevoerd. Uw programma loopt verder terwijl de muziek speelt! Dit wordt in het onderstaande voorbeeld gedemonstreerd. Als uw programma eindigt voordat de muziek is afgelopen, wordt de muziek ook afgebroken. Alleen de laatst gespeelde klank blijft en moet uitgeschakeld worden met VOL 0 of WAVE st,00000000. De in het handboek beschreven instructie PLAY 0 heeft geen effect. PLAY 2 kan ook zeer goed gebruikt worden, als u tijdens de muziek verschillende klankparameters wilt veranderen, zoals in de voorbeelden bij VOL en WAVE. Helaas kunt u niet controleren, wanneer het programma ten einde is. Daarom moet u de timimg van uw programma steeds door proberen bepalen. Laten we het voorbeeld maar eens bekijken.

```
100 REM ***************************
110 REM **                       **
120 REM ## MUSIC/PLAY-VOORBEELD **
130 REM **                       **
140 REM ***************************
150 REM
160 REM
170 REM ***************************
180 REM **                       **
190 REM **  ADE ZUR GUTEN NACHT  **
200 REM **                       **
210 REM ***************************
220 REM
230 VOL 15 : REM VOLUME
240 REM
250 REM NOTEN :
260 REM
270 A$ = "S1c3Gf3Hg3GA3Ga3Ga3Fg3Ff3Gg3Ga3HA3Gd4Gc4Gc4FA3Fa3G
280 A$ = A$ + "c4Gc4GA3Fa3FA3Gc4Ga3HSgh"
290 B$ = "c4Ga3Hc4Gf4Gd4Gd4Fc4Fa3GA3Gg3HA3Gd4Gc4G"
300 B$ = B$ + "c4FA3Fa3Gc4Gc4GA3Fa3FA3Gc4Ga3HGSgE"
310 REM
320 REM
330 REM DRIEHOEK
340 WAVE 1,00010001
350 ENVELOPE 1,4,5, 5, 9
360 REM *********************
370 MUSIC 10,A$ + B$ + B$
380 PLAY 2
390 REM *********************
400 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
410 PRINT"ADE ZUR GU-TEN NACHT,"
420 FOR W=1 TO 2600 : NEXT W
430 PRINT"JEZT WIRD DER SCHLUSS GEMACHT,"
440 REM
```

```
450 FOR W=1 TO 2600 : NEXT W
460 PRINT "DASS ICH MUSS SCHEI-DEN."
470 FOR W=1 TO 2600 : NEXT W
480 FOR X=1 TO 2
490 PRINT
500 PRINT "IM SOMMER, DA WAECHST DER KLEE,"
510 FOR W=1 TO 2600 : NEXT W
520 PRINT "IM WINTER, DA SCHNEIT'S DEN SCHNEE,"
530 FOR W=1 TO 2600 : NEXT W
540 PRINT "DA KOMM ICH WIE-DER."
550 FOR W=1 TO 2600 : NEXT W
560 NEXT X
```

Ik hoop, dat het niet te moeilijk voor u is, om de afzonderlijke letters in de strings voor de definitie van de klankvolgorde te begrijpen. In het algemeen wordt hier naar de uiteenzettingen bij VOL (paragraaf 15.2) verwezen. Hier worden nog een keer kort de belangrijkste zaken herhaald. Het is belangrijk, dat u zich in de klein/groot tekstmodus bevindt. Kleine letters zijn normale noten en worden door een gewone lettertoets ingevoerd. Ze verschijnen eveneens als kleine letters op het beeldscherm. Hoofdletters betekenen in deze listing controletekens. Dit zijn de 8 functietoetsen (E-L) en de SHIFT CLR/HOME - invoer (S). Ze verschijnen op het scherm als inverse hoofdletters. Cursief gedrukte hoofdletters betekenen verhoogde noten en worden aangegeven door de intoetsing van de betreffende lettertoets samen met SHIFT. Ze verschijnen als normale hoofdletters op het tekstscherm. Getallen worden heel normaal ingevoerd en bepalen de stem of de octaven.

15.7 Testopdrachten

1. Wat wordt door de instructie VOL 2 bewerkstelligd?

   A) De geluidssterkte voor alle stemmen wordt op stand 2 gezet.
   B) De geluidssterkte voor stem 2 wordt bepaald.
   C) De syntax is niet juist.
   D) VOL 2 is de op twee na hardste instelling.

2. Waar moet u bij WAVE op letten?

A) WAVE bepaalt de golfvorm voor alle stemmen
B) Ruis mag niet met een andere golfvorm gecombineerd worden
C) De syntax luidt: WAVE 2,%00010001

3. Hoe moet u de instructie MUSIC gebruiken?

A) Die instructie bestaat helemaal niet
B) Eerst wordt de tijd van de maat ingevoerd, daarna de volgorde van de klanken
C) De omgekeerde volgorde van B)
D) Een klank wordt gedefinieerd doordat u de naam van de klank, het octaaf en daarna de waarde van de klank invoert
E) Verlaagde klanken worden ingevoerd, doordat u de naam van de klank SHIFT.

16 STURENDE RANDAPPARATUUR

Uw Commodore 64 heeft verschillende mogelijkheden om sturende elementen als bouwstenen van randapparatuur aan te sluiten. Hiervoor bevinden zich aan de rechterkant van uw computer twee zogenaamde controlports. Dat zijn contacten voor de aansluiting van joystick, paddle, lightpen of zelfs van zelfgemaakte stuureenheden. De indeling en functies van alle pinnen van die aansluitmogelijkheden worden in uw CBM 64-gebruikershandboek beschreven. U hoeft dit niet absoluut te weten om bijvoorbeeld een joystick op uw apparaat aan te sluiten en te gebruiken. Daarom slaan we een nadere uitleg over. De afzonderlijke apparaten, die uw computer en ook Simon's BASIC helpen, worden hier kort beschreven en de betreffende instructies verklaard.

16.1 Lightpen

Onder lightpen verstaat men een penvormig voorwerp, dat voor de invoer of bepaling van een punt op het beeldscherm dient en het direkte contact tussen u en de televisie mogelijk maakt. Met de lightpen is het dus mogelijk door een simpele aanraking van het beeldscherm met de punt van de pen aan de computer een beeldschermpositie te geven. Hoe gaat dit in z'n werk? U wijst met uw in controlport 1 gestoken lightpen op een punt van het beeldscherm. Daarbij maakt het niet uit, of dit punt zich binnen of buiten het eigenlijke tekstvenster bevindt. De computer is dan in staat dit punt te identificeren, hij kent dus de coördinaten van dit punt. Als u deze in uw programma opvraagt, kunt u bijvoorbeeld vaststellen, of er zich op die plaats een bepaald object bevindt. Of u tekent precies op die plaats een punt en nog een en nog een... zodat u op het beeldscherm kunt tekenen! Een ander idee zou zijn, om de lightpen als een gemakkelijke besturing van de cursor te gebruiken. Er zijn een heleboel mogelijkheden voor het gebruik. Eerst moeten er nog een paar dingen over het coördinatensysteem gezegd worden, omdat bij het gebruik van de lightpen verschillen met de ons bekende grafiekindeling opduiken. Tegelijkertijd bestaan er vaak van televisie tot televisie verschillen met betrekking tot de randwaarden. De hier aangegeven richtwaarden kunnen en moeten dus door u gecorrigeerd worden. Er wordt vanuit de rand van het beeldscherm en niet vanuit de rand van het tekstvenster gemeten. Daarbij ligt de linkerrand ongeveer bij x=30. De x-coördinaat van de rand van het linker

tekstvenster luidt x=40, die van de rechterrand 200. Het eind van het beeldscherm aan de rechterkant ligt op x=210. In de y-richting, gemeten vanaf de bovenkant van het beeldscherm, is dit nog iets gecompliceerder. Net als bij de x-indeling beginnen de y-coördinaten bij y=30. De bovenrand van het tekstvenster ligt bij y=40 en de onderkant bij y=240. De onderste rand van het beeldscherm wordt echter in tweeën gedeeld. Tot y=255 is het verloop van de coördinaten nog regelmatig. Maar halverwege de breedte op deze hoogte wordt y=0 om uiteindelijk helemaal op "de bodem" te eindigen met y=25.

Om dus de lightpencoördinaten om te rekenen op de voor ons vetrouwde indeling gebruiken we de volgende formule:

```
x = (xp-40)*2
y = yp-40
```

waarbij x en y de grafiek- en xp en yp de lightpen-coördinaten zijn. In de x-richting hebben deze dus de halve oplossing, wat echter niet veel uitmaakt, omdat de lightpen van nature toch al enigszins in de x-richting verstrooit. Om door leesfouten verkeerde negatieve of te grote waarden in deze formule te vermijden, moet u in een BASIC-regel direkt na het inlezen de verkregen waarden testen, zoals dat ook in het voorbeeld wordt gedaan.

16.1.1 PENX

Format	PENX
Parameter	----
Voorbeeld	X = PENX
Functie	Bepalen van de x-coördinaat van de lightpen

Om de coördinaten van de actuele positie van de lightpen op het beeld vast te stellen, zou er een hele reeks gecompliceerde stappen met PEEKs en POKEs noodzakelijk zijn, die met de verschillende registers van de VIC samenhangen. Simon's BASIC echter maakt het u hierbij heel gemakkelijk. De instructie PENX maakt het u mogelijk, zonder omwegen, de x-coördinaten van de lightpen te weten te komen. PENX wordt evenals TEST, LIN, of SQR als een functie gebruikt. U kunt PENX dus in variabelen opslaan en er berekeningen mee maken:

x = 5 * PENX + 4

Daarbij heeft deze instructie steeds de laatste positie van de pen als waarde. Verschillend met de oude vertrouwde manier om coördinaten te bepalen, met het nulpunt in de linker bovenhoek van het tekstvenster, wordt bij PENX de afstand van het punt vanaf de linkerrand van beeldscherm plus kader aangegeven. Desondanks is de oplossing, dus het maximale aantal punten in een regel, veel kleiner dan we gewend zijn bij de grafiek met het hoogoplossend vermogen. De indeling van de punten is wat gecompliceerd en wordt in de inleiding van dit hoofdstuk uitgelegd.

16.1.2 PENY

Format	PENY
Parameter	---
Voorbeeld	Y = PENY
Functie	Bepalen van de y-coördinaat van de lightpen

PENY geeft op dezelfde manier als in paragraaf 16.1.1 beschreven, de y-coördinaten van de actuele (of laatste) beeldschermpositie van de lightpen aan. Weer wordt die direkt vanaf de rand van het beeldscherm gemeten. Deze keer begint de telling bij y=30 aan de bovenrand. Van boven naar beneden zijn er in totaal ongeveer 250 punten. Als u de betreffende coördinaten in de grafiek van het Simon's BASIC wilt hebben, moet u ze eerst omrekenen.

Voorbeeld:

Dit voorbeeld maakt het mogelijk om direkt met de lightpen op het grafiekbeeldscherm te tekenen. Probeert u het maar. U kunt dus nu direkt uw handtekening zetten of tekeningen maken.

```
100 REM *************************
110 REM **                     **
120 REM **  PENX/PENY-VOORBEELD **
130 REM **                     **
140 REM *************************
150 REM
```

```
160 HIRES 6,7
170 PROC PLOT: REM SYMBOLISCH SPRONGADRES
180 X = (PENX - 40)*2 : REM X-COORDINAAT BEREKENEN
190 Y = PENY - 40 : REM Y-COORDINAAT BEREKENEN
200 IF X<0 OR X>319 OR Y<0 OR Y>199 THEN CALL PLOT
210 LINE XA,YA,X,Y,1 : REM LIJN TEKENEN VAN OUDE NAAR NIEUWE
    COORDINATEN
220 XA = X
230 YA = Y : REM WAARDE VASTLEGGEN
240 GET A$ : REM TOETS HALEN
250 IF A$="=" THEN HIRES 6,7 : REM BIJ "=" WISSEN
260 CALL PLOT
```

16.2 Paddle en Joystick

We komen nu bij een soort besturingsapparaatjes, dat graag bij spelletjes ingezet wordt, maar ook voor vele andere doeleinden wordt gebruikt. Allereerst iets over de paddles. Uw CBM 64 heeft in totaal 4 zogenaamde A/D-omzetters. Dit zijn elektronische schakelingen, die het mogelijk maken een analoge waarde om te zetten in een digitale waarde. Een analoge waarde verloopt geleidelijk in plaats van stapvormig, bijvoorbeeld de temperatuur. Een digitale waarde is een waarde die in een getal kan worden uitgedrukt. Deze digitale waarde is onderdeel van een scala van veel mogelijke waarden (hier 256) waarmee men kan rekenen. Zoals gewoonlijk worden aan deze A/D-converters, die eveneens door de controlport te bereiken zijn, paddles aangesloten. Dit zijn potentiometers, waarmee u waarden van 0-255 kunt instellen door naar links of naar rechts te draaien. Deze waarden kunnen dan in een speciaal programma gebruikt worden. Bijvoorbeeld bij de besturing van de verschillende parameters van de geluidsinstructies, waardoor u fraaie effecten kunt bereiken.

De joysticks zijn de bekendste randapparaatjes, omdat ze vooral voor spelletjes uitstekend geschikt zijn. Joysticks zijn echte stuurknuppeltjes, die u in alle richtingen kunt trekken en duwen. Meestal hebben ze ook een drukknop. U kunt vragen in welke richting de joystick wijst en of er wel of niet op de knop gedrukt is. Zo kunt u objecten over het beeldscherm laten bewegen of schoten afvuren. Zo goed als alle spelletjes voor de CBM 64 worden met deze knuppeltjes gespeeld, die natuurlijk ook uitstekend geschikt zijn als besturing van de cursor.

16.2.1 POT

Format	POT (p)
Parameter	p : legt het paddlenummer vast
Voorbeeld	X = POT (1)
Functie	Aflezen van de plaats van de paddle

POT is de adequate instructie om de actuele waarde van de paddle vast te stellen, dus de plaats van de met p aangegeven paddle. POT wordt ook als een normale functie bij berekeningen ingebouwd. De toepassing hoeven we niet te beschrijven, omdat we daarop al bij de PEN-instructies zijn ingegaan.

Voorbeelden:

```
100 REM **********************
110 REM **                  **
120 REM **  POT-VOORBEELD 1 **
130 REM **                  **
140 REM **********************
150 REM
160 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
170 P = POT(0) : REM PADDLEWAARDE UITLEZEN
180 PRINT P
190 GOTO 170
```

```
100 REM **********************
110 REM **                  **
120 REM **  POT-VOORBEELD 2 **
130 REM **                  **
140 REM **********************
150 REM
160 PRINT CHR$(147) : REM BEELDSCHERM WISSEN
170 P0 = POT(0) /  6.4 : REM PADDLEWAARDE 0 UITLEZEN
180 P1 = POT(1) / 10.3 : REM PADDLEWAARDE 1 UITLEZEN
190 PRINT AT(P0,P1) "*" : REM OP AANGEDUIDE PLAATS TEKENEN
200 GOTO 170
```

16.2.2 JOY

Format	JOY
Parameter	----
Voorbeeld	J = JOY
Functie	Uitlezen van de joystick-instelling

Op soortgelijke manier als bij vorige instructies kunt u er met de JOY-instructie achter komen in welke richting de joystick werd geduwd. Vreemd genoeg geldt dit slechts voor de joystick aan port 2. Port 1 is waarschijnlijk van deze instructie uitgesloten omdat bepaalde joystick-instellingen hier letters en andere tekens naar de computer sturen. De instructie geeft voor elk van de acht richtingen een bepaalde waarde aan met de 0 in het midden en de 1 recht omhoog:

8	1	2	
7	0	3	
6	5	4	128

Staat de joystick recht omhoog, dan levert de functie JOY een 1, schuin links omhoog een 8, enzovoort. Als tegelijkertijd de drukknop wordt bediend, wordt de uitgegeven waarde met 128 verhoogd.

Voorbeeld:

```
100 REM ********************
110 REM **                **
120 REM **  JOY-VOORBEELD **
130 REM **                **
140 REM ********************
150 REM
160 IF JOY=1 OR JOY=129 THEN PRINT "BOVEN"
170 IF JOY=3 OR JOY=131 THEN PRINT "RECHTS"
180 IF JOY=5 OR JOY=133 THEN PRINT "BENEDEN"
190 IF JOY=7 OR JOY=135 THEN PRINT "LINKS"
200 IF JOY>=128 THEN PRINT "VUURKNOP"
210 GOTO 160
```

17 AANHANGSEL

17.1 Oplossingen van de testopdrachten

In de volgende regels vindt u de oplossingen van de testopdrachten die in dit boek zijn gegeven.

paragraaf	oplossingen				
2.1.3	1) a	2) c	3) c	4) c	
2.2.4	1) a	2) a	3) b,c	4) b,c	
2.3.5	1) b	2) a	3) c	4) b	
	5) b	6) b			
3.1.6	1) c	2) d	3) a,b	4) a	5) b
3.2.5	1) a	2) d	3) c	4) b	5) c
5.1.5	1) b	2) b	3) b	4) a	5) d
5.2.6	1) a,b	2) a	3) b	4) b	5) b
6.3	1) b	2) b	3) a,b,c		
7.1.5	1) b	2) a	3) b	4) c	5) b
7.2.3	1) c ?	2) a,c	3) a	4) b,d	
8.5	1) c	2) a,c	3) d	4) c	5) c
	6) c	7) b	8) a	9) c	
9.1.6	1) b	2) b,d,f	3) c,d		
9.2.5	1) c	2) c			
9.3.4	1) b,d	2) c			
9.4.6	1) c	2) b,c	3) c		
9.5.5	1) d	2) a,d			
10.3	1) d	2) c	3) b	4) b	
11.2.3	1) a,b	2) c	3) b		
12.2.7	1) b	2) c	3) b,c		
12.3.10	1) c,e,f	2) b	3) a,b,c		
12.4.3	1) a,c	2) c	3) a,d		
12.5.3	1) b,c	2) b			
13.5	1) b,c,d	2) b			
14.1.5	1) b,c	2) d	3) b,d		
14.2.6	1) b,d	2) a			
15.7	1) a	2) b	3) b,d,e		

17.2 Kleurtabel

Bij de verschillende grafiekinstructies moet u de kleur waarmee u wilt tekenen in de vorm van een kleurcode invoeren. Elk van de 16 te kiezen kleuren heeft zo'n code. U vindt er hier een lijst van. Tegelijk wordt de toetscombinatie genoemd waarmee u door intoetsen de tekstkleur kunt besturen. De vermelde toetsen moeten gelijktijdig worden aangeslagen.

```
CODE..KLEUR.....TOETSEN----CODE..KLEUR.....TOETSEN
  0...zwart.....k 1          8...oranje......c 1
  1...wit.......k 2          9...bruin.......c 2
  2...rood......k 3         10...lichtrood...c 3
  3...cyaan.....k 4         11...grijs.1.....c 4
  4...paars.....k 5         12...grijs.2.....c 5
  5...groen.....k 6         13...lichtgroen..c 6
  6...blauw.....k 7         14...lichtblauw..c 7
  7...geel......k 8         15...grijs.3.....c 8
```

17.3 Foutmeldingen

Hier zijn alle foutmeldingen van Simon's BASIC die worden toegevoegd aan de bekende foutmeldingen van het standaard bedrijfssysteem. In het met Simon's BASIC meegeleverde handboek ontbreken ze helaas.

PROC NOT FOUND
: wordt gegeven als een symbolisch sprongadres wordt aangestuurd, dat u niet als zodanig in uw programma hebt gedefinieerd.

INSERT TOO LARGE
: onstaat als u bij gebruik van de instructies INST of INSERT een karakterpositie opgeeft die niet bestaat of als een op te nemen string groter is dan de opnemende string.

STRING TOO LARGE
: verschijnt als de resulterende string bij gebruik van

INST en INSERT het aantal van 255 karakterposities overschrijdt.

NOT BINARY CHAR
geeft aan dat u bij de vorming van een binair getal achter % in plaats van een 1 of een 0 iets anders hebt ingetypt. De fout treedt ook op als u % ten onrechte gebruikt, bijvoorbeeld achter de instructie WAVE.

NOT HEX CHAR geeft aan dat u na $ geen hexadecimale digit hebt ingevoerd.

END PROC WITHOUT EXEC
spreekt voor zichzelf als u een niet bestaande subroutine wilt beëindigen met ENDPROC en is vergelijkbaar met RETURN WITHOUT GOSUB ERROR van standaard BASIC.

END LOOP WITHOUT LOOP
hetzelfde als NEXT WITHOUT FOR ERROR in standaard BASIC als in het programma een END LOOP wordt gevonden die niet voorheen met LOOP werd geopend.

UNTIL WITHOUT REPEAT
eenzelfde geval in een REPEAT-lus

STACK TOO LARGE
geeft aan dat een te grote nesteling van lussen en subroutines in uw programma optreedt. U zult moeten vereenvoudigen.

BAD CHAR FOR A MOB
signaleert verkeerde tekens, of te veel, of te weinig in een sprite- of karakterdefinitie. Ze duikt ook op als u te weinig regels in de definitie hebt gebruikt of tussen de DESIGN-instructie en de definitie een BASICregel hebt ingevoegd.

BAD MODE is een universele foutmelding, evenals ILLEGAL QUANTITY ERROR, die een overschrijding van het bereik bij Simon's BASIC aangeeft.

17.4 Alfabetisch overzicht van de instructies

@	240
$	83
%	82
ANGL	210
ARC	205
AT	106
AUTO	18
BFLASH	114
BFLASH 0	115
BLOCK	198
CALL	68
CENTRE	103
CGOTO	70
CHAR	227
CHECK	268
CIRCLE	201
CMOB	257
COLD	41
COPY	163
CSET 0/1	244
CSET 2	177
DELAY	30
DESIGN 0/1	251
DESIGN 2	239
DETECT	267
DIR	157
DISAPA	55
DISK	156
DISPLAY	16
DIV	77
DOWN	147
DRAW	219
DUMP	40
DUP	90
END PROC	67
ENVELOPE	281
ERRLN	44
ERRN	44
EXEC	69
EXOR	80
FCHR	129
FCOL	132
FETCH	150
FILL	135
INV	138
JOY	294
KEY	13
LEFT	141
LIN	108
LINE	191
LOCAL	73
LOOP/EXIT IF/END	
LOOP	63
LOW COL	178
MEM	236
MERGE	23
MMOB	259
MOB OFF	261
MOB SET	254
MOD	76
MOVE	136
MULTI	173
MUSIC	283
NO ERROR	46
OFF	13
OLD	42
ON ERROR	44
OPTION	29
OUT	52
PAGE	28
PAUSE	109
PAINT	214
PENX	296
PENY	291
PLACE	90
PLAY	285
PLOT	186
POT	293
PROC	66
RCOMP	60
REC	194
REPEAT/UNTIL	61
RENUMBER	20
RESET	153
RETRACE	39
RIGHT	143
RLOCMOB	264
ROT	223

FIND ... 31
FLASH ... 111
FRAC ... 79
GLOBAL ... 74
HI COL ... 181
HIRES ... 171
HRDCOPY ... 163
IF/THEN/ELSE ... 58 151
INKEY ... 151
INSERT ... 86
INST ... 88
SCRLD ... 161
SCRSV ... 159
SECURE 0 ... 56
TEST ... 187
TEXT ... 229
TRACE ... 37
UP ... 145
USE ... 105
VOL ... 275
WAVE ... 277

17.5 Simon's BASIC gegevens

- Beslaat 16 Kbyte van het BASIC-geheugen ($7FFF-$9FFF).
- Beslaat gedeeltelijk ruimte in het geheugenbereik onder het BASIC-ROM.
- Schakelt gedeeltelijk op BASIC-ROM en BASIC-RAM.
- Geen RUN/STOP-uitwerking met POKE788,52 en evenmin RUN/STOP-RESTORE met POKE792,193. De interrupt-vector is afgebogen.
- Loopt met DATA BECKER IEC-bus versie S3.0 (zonder verandering van de karakterset).
- Loopt met DISCOMAT (SUPERTWIN en DISKMONITOR) als daarbij de modus "SUPERTWIN met IEC-bus" wordt gekozen.
- Loopt niet met de 80-tekens kaart MAXI 64.
- Loopt met PROFI-MON 64 uit het Assemblerpaket PROFIMAT.
- Geen mogelijkheid tot hardcopy met de Commodore printer VC-1526.

Simon's Basic is een hit – indien men het juist weet te gebruiken. Dit trainingsboek verklaart u de omgang met de meer dan 100 instructies van Simon's Basic. Alle bevelen worden uitvoerig beschreven, ook die, welke niet in het handboek staan! Natuurlijk tonen de auteurs ook de gebreken van Simon's Basic en geven belangrijke tips, hoe men deze kan omzeilen.

Uit de inhoud:
Inleiding in het CBM-BASIC 2.0 ★ programmeerhulp ★ foutenbehandeling ★ programmabescherming ★ programmastructuur ★ variabelen ★ getallenbehandeling ★ invoercontrole ★ opgave en uitvoering van instructies voor randapparatuur ★ grafiek ★ karaktersetvervaardiging ★ sprites ★ muziek.
Vanzelfsprekend bevat dit boek veel voorbeeldprogramma's en interessante programmeertrucs. Na ieder hoofdstuk testopgaven voor de optimale zelfstudie en voor toetsing van de verworven kennis.

ISBN 90 229 3327 X

Commodore Bibliotheek 5

DATA BECKER
NEDERLANDS *